VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

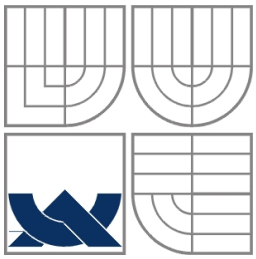
OVLÁDÁNÍ ROBOTICKÉHO MANIPULÁTORU MIKROKONTROLÉREM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

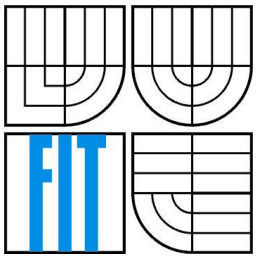
AUTOR PRÁCE
AUTHOR

Bc. MARTIN ZEMÁNEK

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

OVLÁDÁNÍ ROBOTICKÉHO MANIPULÁTORU MIKROKONTROLÉREM

ROBOTIC MANIPULATOR CONTROLLING USING MICROCONTROLLER

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MARTIN ZEMÁNEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. RICHARD RŮŽIČKA, Ph.D.

BRNO 2007

Abstrakt

Tato diplomová práce pojednává o ovládání robotického manipulátoru ROB 1-3 pomocí běžného joysticku a mikrokontroléru HC08 NITRON a LJ12. Jejím cílem je vytvoření modulu pro ovládání výchylek servomechanismů robota zpracováním analogového signálu z připojeného joysticku.

V teoretických úvodních částech jsou uvedeny poznatky o rozhranní PC Gameport, architektury modelářských serv a především vlastnosti mikrokontrolérů řady HC08. Na tuto část navazují návrhy možného vzájemného propojení jednotlivých komponent, jak pro autonomní, tak pro verzi připojitelnou na laboratorní vývojový kit, na kterých by měly stavět návrhy programového řízení.

Praktickou část zastřešují konkrétní realizované návrhy implementace jak fyzického zapojení modulu, tak programy pro řídící činnost mikrokontrolérů řady HC08. Jsou uvedeny omezující kritéria v analýzách a celkové shrnutí výsledku práce v závěrečném vyhodnocení návrhů.

Klíčová slova

mikrokontrolér, HC08, gameport, joystick, servomechanismus, robot, ROB1-3, manipulátor, ovládání

Abstract

This master's thesis deals with control robotic manipulator ROB 1-3 using common joystick and microcontroller HC08 NITRON and HC08 LJ12. The goal of this work is creation module for controlling axes position of servos in robot by analog signal coming from connected joystick.

There are mentioned knowledge about PC Gameport interface, architecture of servos and above all parameters of microcontrollers of HC08 family in the theoretical introduction. Upon this part there are formed possible suggestions about connecting individual parts together for both autonomous and version using development kit. Programming code suggestion should be based on these connections during further development.

Practical part contains concrete designs of implementation both board wiring and programs that operates on microcontroller HC08. The limitations of using it are described in analysis and final results are shown in statistics results.

Keywords

microcontroller, HC08, gameport, joystick, servo, robot, ROB1-3, manipulator, control

Citace

Zemánek, M.: Ovládání robotického manipulátoru mikrokontrolérem , diplomová práce, Brno, FIT VUT v Brně, 2007

Ovládání robotického manipulátoru mikrokontrolérem

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením
Ing. Richarda Růžičky, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Martin Zemánek
22.5.2007

Obsah

1 Úvod

Výpočetní systémy, reprezentované v dnešní době především osobními počítači vybavenými různými procesory a přidanými periferiemi, jsou schopny díky své univerzálnosti a modulárnosti ovládat přímo řadu vstupně-výstupních linek, ale za cenu degradace výkonu systému a možné nespolehlivosti. Pro všechny tyto aplikace, kdy je opakovaně prováděna operace vyžadující dodržení určitých parametrů vstupů resp. výstupů jako je čas, rychlost, propustnost, atd., je nutné navrhnout vlastní systém řízení, který je schopen autonomně a pouze pro danou úlohu efektivně pracovat. Takovýto systém je buď realizovaný jako složitý obvod analogových a číslicových prvků nebo jako vestavěný systém řízený mikrokontrolérem nebo programovatelným polem.

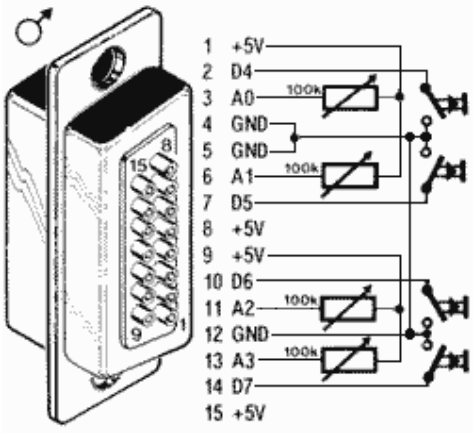
Mé dřívější experimentování s různými zařízeními připojovanými k počítači a jejich ovládání přes programovaní portů, které ne vždy bylo schopno samo o sobě zajistit správnou činnost a bylo potřeba dořešit neefektivní modul řízení, bylo důvodem, proč jsem si vybral jako téma své diplomové práce ovládání robotického manipulátoru mikrokontrolérem. Využití mikrokontroléru jako „srdce“ aplikace dává prostor k vytváření více variant řízení při ponechaní existujících spojů a zmenšení rozměrů výsledného modulu.

Cílem práce je navržení modulu pro ovládání robotického ramena, v podání robota ROB 1-3, běžným analogovým joystickem. Na začátku budou poskytnuty informace o parametrech a vlastnostech analogového rozhraní PC Gameport pro joysticky. Na teoretický úvod analogového rozhraní PC Gameport navazuje kapitola o řízení modelářských servomechanismů a pak následují kapitoly o mikrokontrolérech HC08, jako stěžejního prvku celé aplikace. Jádrem práce bude objasnění analýz a návrhů konkrétních modulů pro řízení manipulátoru a zdrojových programů v jazyku C pro demonstraci funkčnosti řešení. Budou objasněna jednotlivá východiska a úskalí, která se v průběhu vývoje vyskytla.

2 Rozhraní PC Gameport

Rozhraní PC game port je analogové rozhraní pro připojení herních zařízení k osobním počítačům, která představila firma IBM s prvními počítači. Umožňovalo sledovat 4 analogové osy a 4 tlačítka pro jeden port, na každý port až dva joysticky nebo čtyři pedály za použití speciální rozdvojky („Y“). Cílem bylo vytvoření levného a jednoduchého rozhraní s použitím pouze obvodů pro komunikaci po sběrnici (zpočátku ISA) a čtyř obvodů pro *AD („analog-to-digital“)* konverzi. S použitím speciálních kabelů je možné připojit také zařízení ve standardu Roland MPU-401 *MIDI* (*Musical Instrument Digital Interface*) rozhraní pro vstup elektrických hudebních zařízení. I přes snahy o zlepšení parametrů tohoto rozhraní, je v dnešní době toto rozhranní vytlačováno rozšiřujícím se sériovým rozhraním *USB* („Universal Serial Bus“). Přesto i nadále jsou levnější počítače a zvukové karty vybavovány tímto rozhraním, které může v kombinaci s dalšími logickými prvky sloužit např. k měření napětí a proudu.

2.1 PC Gameport



Obrázek 2.1.1: Zapojení konektoru PC Gameport

2.1.1 Popis signálů

Rozhraní PC Gameport je vyvedeno na konektor D-SUB s 15 vývody v provedení male *viz obr. 2.1.1*. Zapojení vstupů a výstupu ukazuje *tabulka 2.1.1.1*.

Vývod	Označení	Směr	Popis
1	+5V	výstup	Napětí 5V DC
2	B1	vstup	Tlačítko 1
3	X1	vstup	Joystick 1 - osa X
4	GND	-	Zem
5	GND	-	Zem
6	Y1	vstup	Joystick 1 - osa Y
7	B2	vstup	Tlačítko 2
8	+5V	výstup	Napětí 5V DC
9	+5V	výstup	Napětí 5V DC
10	B4	vstup	Tlačítko 4
11	X2	vstup	Joystick 2 - osa X
12	GND	-	Zem
13	Y2	vstup	Joystick 2 - osa X
14	B3	vstup	Tlačítko 3
15	+5V	výstup	Napětí 5V DC

Tabulka 2.1.1.1: Popis signálů PC Gameport

2.1.2 Komunikace

Typická implementace rozhraní game portu obsahuje čtyři, různým způsobem navržené, multivibrátory, které převádějí hodnoty ze vstupů X1, X2, Y1, Y2 na číselnou reprezentaci vyčítanou přes sběrnici počítače do obslužného programu. Ze vstupů není patrné, kdy má být hodnota předána ke zpracování, proto je správné časování a resetování multivibrátoru nutnou podmínkou pro zajištění správných hodnot. S vzrůstající rychlostí zpracování byl tento problém odstraněn použitím vlastního řízení snímaní hodnot.

Analogová hodnota na vstupech z joysticku je snímána klopným obvodem např. monostabilním multivibrátorem s malým kondenzátorem, který je nabit napětím procházejícím potenciometrem. Délka pulsů generovaná tímto způsobem představuje hodnotu výchylky v jedné ose joysticku. Přesnou činnost obvodu je možné rozdělit do 4 kroků:

1. V případě, že je zařízení v klidu, je kondenzátor plně nabit a na výstupu je log.1
2. Pokud je multivibrátor resetován (zápisem na adresu 201h), vybije kondenzátor a vrátí hodnotu log. 0.
3. Kondenzátor se začne nabíjet přes odpor potenciometru.

4. Jakmile dosáhne dané úrovně napětí, vrátí zpět hodnotu log. 0.

Tento způsob převodu analogové hodnoty na číslicovou je v tomto podání typický pro počítačové systémy. V případě této práce není AD převod řešen tímto způsobem, ale pouze odečtem číselné hodnoty napětí na řídícím členu s použitím 10kΩ odporu.

2.2 Joystick

Je periferní zařízení PC skládající se z pákového ovladače, měřící výchylku ve dvou nebo třech osách, a přídavných programovatelných tlačítek. Výstupem je absolutní hodnota výchylky vůči nulové definované pozici dané osy, která je vyjádřena odporem u analogového joysticku nebo přepočítanou číselnou hodnotou u moderních digitálních joysticků. Získané hodnoty jsou následně využity pro změnu pozice virtuálního objektu na zobrazovacím zařízení nebo jako vstupní parametry předávané robotům.

Analogové joysticky jsou typicky složeny z dvou potenciometrů s proměnnou hodnotou odporu v rozmezí 0 Ω až 100 kΩ. Jednotlivé potenciometry jsou připojeny na vstup +5V, střední vývod je připojen k pohyblivému jezdci a třetí přístup je nepřipojen. Převod mechanického pohybu na výchylku potenciometru může být lineární nebo axiální. V případě lineárního převodu mění pohyb páky pozici posuvníku potenciometru v rozsahu 100kΩ, zatímco více používané axiální potenciometry jsou připojeny přímo a využívají pouze omezeného pohybu páky do 60-90 stupňů.

Tlačítka joysticku představují dvoustavový výstup (on/off). Stav tlačítka může být přes interface rozhranní přenášen na vstupy sběrnice. Akce sepnutí tlačítka uzemní daný vývod přes definovanou zem, respektive na daný vývod přivede +5V (log. 1) pomocí přidaných pullup rezistorů.

3 Robotický manipulátor ROB 1-3

Robotický manipulátor ROB 1-3 *obr. 3.1* představuje stavebnici modelu tříosého manipulátoru, který může být řízen osobním počítačem přes speciální modul nebo mohou být řízeny jeho jednotlivé prvky. Robot je polohován modelářskými servomechanizmy Hitec HS-322 ve třech osách – otáčení základny, zdvih ramene a pohyb kleštiny. Jelikož je robot sestaven z plastových dílů obrobených laserem navzájem spojených běžným spojovacím materiálem, není vhodný pro výrobní činnost, ale pouze pro demonstrační a studijní účely.

Obrázek 3.1: Robotický manipulátor ROB 1-3

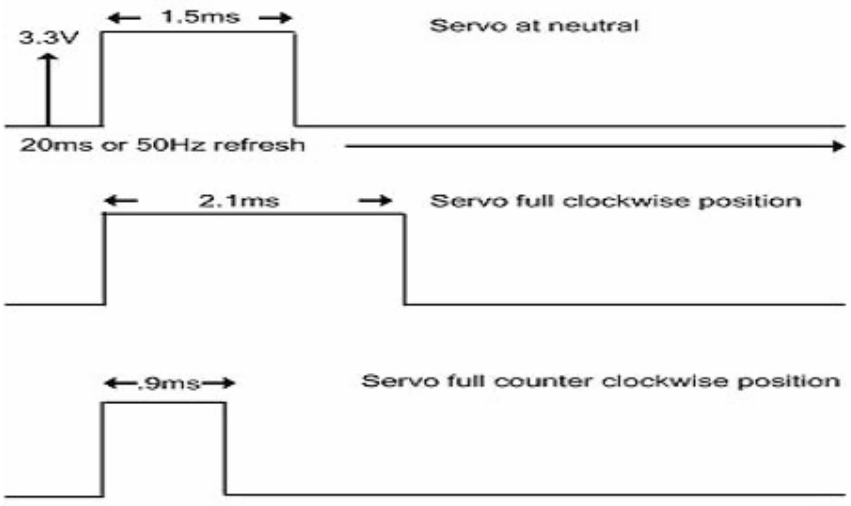
3.1 Servomechanismy Hitec HS-322

Modelářské servomechanismy („serva“) jsou především určeny pro řízení RC („Radio Controlled“) modelů. Mohou být ale také dobře využity pro oblast robotiky, automatizace nebo domácích zařízení. Serva se skládají ze stejnosměrného elektromotorku, vícestupňové převodovky sestavené z ozubených kol, zpětno-vazebného snímače polohy výstupního hřídele a řídící elektroniky. Požadovaná pozice osy serva je nastavována řídící elektronikou pomocí pulsně-šířkové modulace (PWM) modulovaného řídícího signálu. Na vstupu serva je periodicky přiváděn řídící puls, který spouští monostabilní klopný obvod, ten následně generuje impuls o délce odpovídající aktuální poloze serva, ale obrácené polarity než je vstupní řídící puls. Oba tyto pulzy se porovnají a výsledkem je puls, který způsobí natočení elektromotoru jedním nebo druhým směrem.

Servomechanizmus firmy Hitec s označením HS-322 *obr. 3.1.1* patří vlastnostmi mezi standardní analogová „serva“ pro modeláře. S hmotností 43g, velikostí 40 x 20 x 37mm působí silou přes 3 kg při změně pozice do 0.19s. Pří výrobě ozubených částí servomechanizmu byl použit uhlíkový kompozit pro zvýšení odolnosti a životnosti.

Obrázek 3.1.1: Servomechanismus HS-322

Pro správnou činnost je třeba zajistit napájení napětí v rozmezí 4.8 až 6V, tedy hodnot TTL (5V). Pro vstupní signál a přívod napájení je využit konektor typu „S“ s černým vodičem pro zem, červeným pro napájení a žlutým pro vstupní signál. Vstupní signál je tvořen obdélníkovým pulsem 3-4V, ale je možné použít rozmezí TTL o proměnné délce 0.9ms až 2.1ms při kmitočtu 50Hz viz. *obr. 3.1.2*, což odpovídá rozsahu ± 54°. Mechanický rozsah se dá zvětšit až na rozsah ± 90° zvětšením rozsahu řídících impulsů na rozsah 0,5 až 2.5ms. Řídící impulzy šířky, které jsou mimo tento rozsah, mohou způsobit náraz serva na mechanický doraz, a tím jeho poškození.



Obrázek 3.1.2: PWM modulovaný řídící signál

Přestože aplikace uvedeného robotického manipulátoru nevyžaduje rychlé odezvy servomechanismů, je přesto vhodné zajistit co nejpřesnější obnovovací kmitočet („refresh“), jelikož i malé změny ve vstupním signále („jitter“) způsobují „škubání“ serva a výsledkem je větší spotřeba a dřívější opotřebení.

4 Mikrokontroléry řady 68HC08

Rodina mikrokontrolérů, dříve vyráběna pod značnou Motorola, nyní Freescale Semiconductor, je rodina 8-bitových mikrokontrolérů, vycházejících z řady HC05 na bázi mikroprocesoru M6800. Procesory jsou postavené na von Neumanově architektuře s CISC instrukcemi a „big-endian“ paměťovým přístupem.

Svojí výkonností, jednoduchostí, ale i současně širokou paletou modelů patří mezi častá řešení prostorově a výpočetně nenáročných aplikací.

Základní údaje:

- von Neumanova architektura
- minimální počet registrů (A, HX, SP, PC a CCR)
- periférie mapovány do adresového prostoru
- rychlý přístup do paměti jedním ze 16ti adresovacích módů
- rychlost CPU a sběrnice až do 8MHz
- hardwarové násobení a dělení
- podpora vyšších programovacích jazyků
- různé režimy řízení spotřeby

4.1 Značení mikrokontrolérů HC08

Obvody HC08 tvoří ucelenou řadu mikrokontrolérů se stejným CPU, kde se jednotlivé typy liší množstvím periferních obvodů a velikostí pamětí. Pochopení značení tedy hraje důležitou roli pro nalezení nejoptimálnějšího modelu pro zvolenou aplikaci viz. *tabulka 4.1.1*.

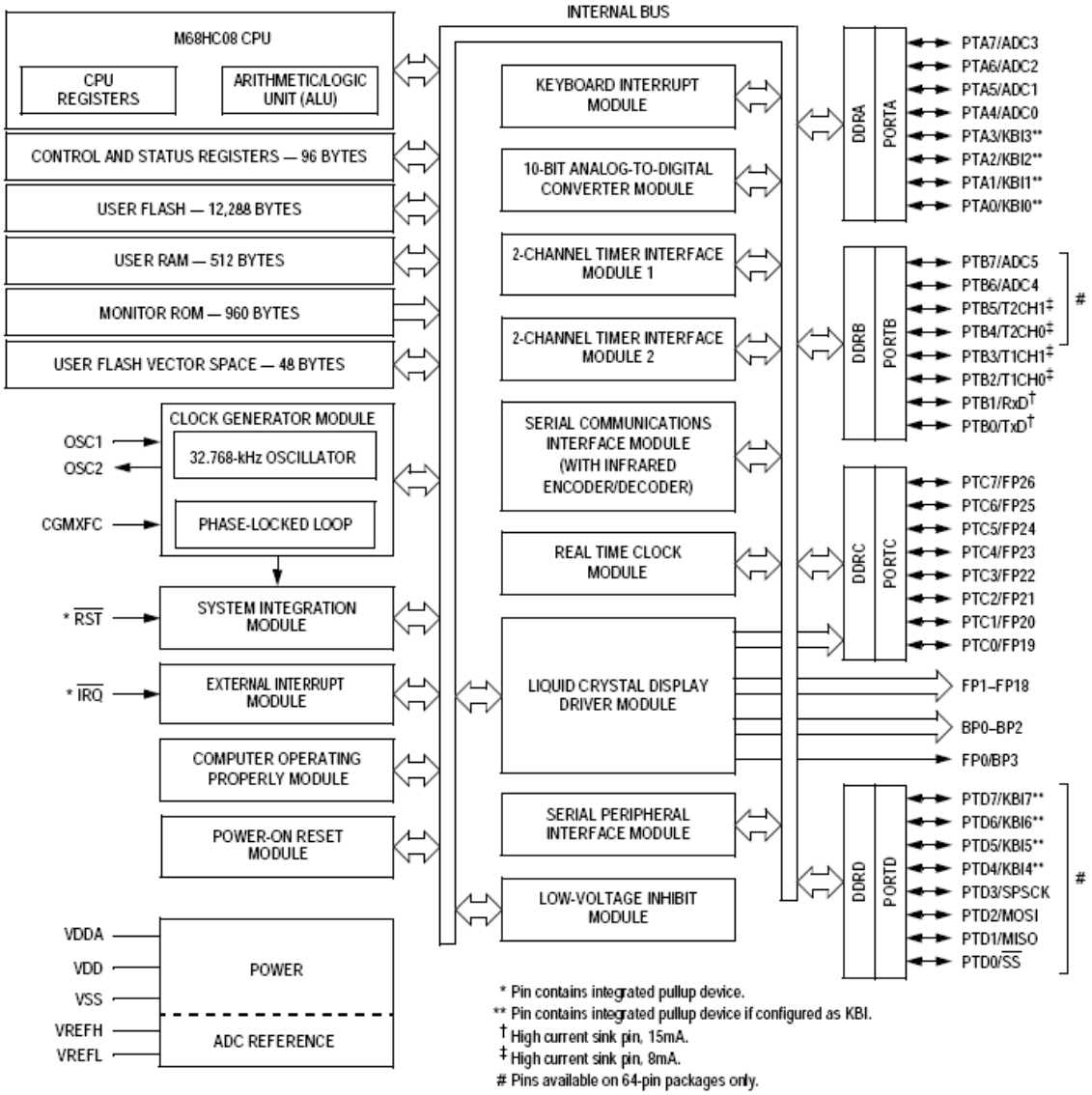
MC	**68**	**HC**	**9**	**08**	**QT**	**4**	**C**	**P**
1	2	3	4	5	6	7	8	9

Tabulka 4.1.1: Značení mikrokontrolérů

1. **MC** – kvalifikovaná součástka
2. **68** – řada mikrokontrolérů
3. **HC** – technologie obvodu
4. **9** – mikrokontrolér s programovatelnou pamětí FLASH, bez označení se identifikuje ROM programová maska

5. **08** – typové označení HC08
6. **QT** – typ v rámci rodiny HC08, popisující skladbu periferií a doporučené oblasti použití
7. **4** – velikost paměti FLASH v kB. Hodnota se pohybuje od 1kB až 64kB
8. **C** – teplotní rozsah
9. **P** – typ pouzdra, P=DIL, DW=SOIC, FA=LQFP

4.2 Blokové schéma HC08



Obrázek 4.2.1: Blokové schéma MC68HC908LJ12

4.3 Režimy činnosti HC08

Mikrokontroléry řady HC08 nabízejí funkci In-System-Programming (ISP), které dovoluje provozovat procesory ve dvou režimech:

- MONITOR MODE
- USER MODE

USER MODE

V tomto režimu běží mikroprocesor při standardním provozu, kdy se provádí kód programu uložený v paměti, jinak také nazývaný uživatelská mód.

MONITOR MODE

Pro podporu ladění aplikací vyvinula firma Motorola, od této řady mikrokontrolérů ,technologii **ISP** a tzv. *režim monitoru*. Tato technologie nahradila dřívější těžkopádné ladění programů, vyžadující další programové a HW prostředky zasíláním příkazů přes sériovou linku (SCI). Důvodem nasazení bylo jednak použití nových pamětí typu FLASH, jednak potřeba jednoduše a spolehlivě programovat již nasazené mikrokontroléry především pro změnu firmwarů.

Pro spojení s hostitelským počítačem byla využita asynchronní sériová komunikace pouze na jediném vodiči (poloduplexně), vývodu mikrokontroléru. Ostatní vstupní/výstupní vývody nejsou tímto režimem ovlivněny, vše je řešeno zcela odděleně. Program, který vykonává režim monitoru, je uložen v paměti ROM. Pro vyloučení vstupu mikrokontroléru do režimu monitoru jsou přesně definovány logické hodnoty na vstupech mikroprocesoru v kombinaci v vyšším napětím na pinu „IRQ“.

4.4 Programovací model HC08

Podstatou programovacího modelu je vytvoření registrové struktury, v rámci které může programátor vytvářet program. Mikrokontroléry HC08 mají následující strukturu:

A 8-bitový střadač, k obecnému použití. Ukládají se do něho především operandy a výsledky aritmetických a jiných operací.

HX 16-bitový indexregistr pro adresování 64kB paměti, rozdělený na registry **H** a **X**

SP 16-bitový ukazatel zásobníku s počáteční hodnotou $00FF, ukazující na první volnou pozici zásobníku. Dá se využít i jako indexregistr.

PC 16-bitový programový čítač obsahující adresu příští načítané instrukce nebo operandu. Po resetu je naplněn vektorem uloženým na adrese $FFFE a $FFFF, první hodnota je adresa první instrukce. Při běhu programu se hodnota automaticky inkrementuje až na případy skoků, kdy je naplněna jinou adresou.

CCR 8-bitový stavový registr obsahující globální masku přerušení a pět příznaků. V –příznak přetečení, H – příznak polovičního přenosu, I – maska přerušení, C – přenos, N – příznak záporného obsahu střadače.

4.5 Mikrokontrolér MC68HC908LJ12

CPU08 - Pracuje na frekvenci 8-MHz(5V) resp. 5-MHz(3.3V) s 512B RAM, 12KB FLASH s podporou ISP, 16 adresovacích režimů, 16-bitový index registr, rychlé dělení a násobení a s podporou paměťových přesunů.

TIM - Dva 16-bitové, dvoukanálové časovače s podporou vstupního/výstupního srovnání a PWM na každém kanále.

RTC - Hodiny reálného času s podporou výběru periody přerušení.

SCI,SPI - Asynchronní, resp.synchronní sériové rozhranní.

ADC - Analogově-digitální převodník 6-kanálový s 10b rozlišením.

IRQ - Vstup vnějšího maskovatelného přerušení s událostí na hranu, úroveň nebo kombinovaně.

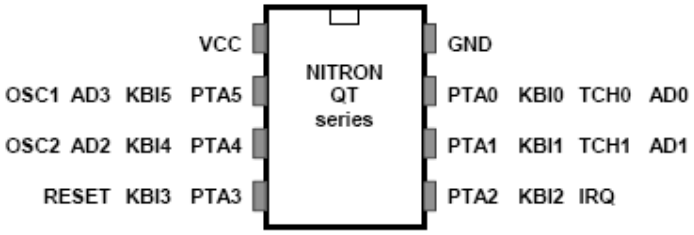
COP - Hlídání správného běhu programu s nezávislým čítačem s vyvoláním resetu nebo přerušení při sníženém napájení, chybné adrese nebo chybné instrukci.

PT - Celkem 32 vstupně/výstupních pinů k obecnému použití s označením PTA, PTB, PTC, PTD.

4.6 Mikrokontrolér MC68HC908QT NITRON

Mikrokontrolér HC08 s označením NITRON patří mezi nejnovější a nejmenší zástupce. Je určen především pro méně náročné, samostatné aplikace, např. řízení otáček, dálková ovládaní nebo jednoduchá rozhraní. Vyrábí se v 8-vývodovém (verze QT) nebo 16-

vývodovém(verze QY) pouzdru DIL nebo SOIC viz *obr. 4.6.1*. Na rozdíl od předešlé verze má pouze 6, resp. 14 vývodů k obecnému použití, disponuje pouze 128B RAM a max. 4kB FLASH. Méně vývodů se odrazilo i na implementaci pouze 4-kanálového 8-bitového AD převodníku nebo 6-ti vstupého klávesnicového rozhraní.



Obrázek 4.6.1: Pouzdro mikrokontroléru NITRON

4.6.1 A/D převodník

Jedná se o 8-bitový A/D převodník, založený na metodě postupné aproximace. Tento typ převodníku vyžaduje pro svou funkci, aby napětí na převodníku zůstalo po celou dobu převodu konstantní. Toho je docíleno pomocí vstupního vzorkovacího zesilovače/multiplexeru. Doba převodu je 16 taktů řídícího kmitočtu A/D převodníku, z nichž je 5 taktů vyhrazeno na vzorkování měřeného vstupního napětí. Řídící kmitočet převodníku je možno měnit v rozmezí 0,5 MHz až 1MHz, což umožňuje maximální vzorkovací frekvenci 62,5 kHz. Dokončení A/D převodu může generovat přerušení s vlastním vektorem přerušení pro jednoduší detekci zdroje přerušení.

4.6.2 Blok rozhranní klávesnice

Další periférií na čipu, která najde své uplatnění v případech, kdy je nutno obsluhovat tlačítka nebo klávesy. Rozhranní generuje přerušení na změnu vybraných bitů na pinech portu PTA (0-5). Výhodou rozhranní je schopnost „probudit“ mikrokontrolér z režimu snížené spotřeby (watt mode, stop mode).

4.6.3 Blok čítačů/časovačů

Je tvořen jednoduchým, volně běžícím 16-bitovým vzestupných čítačem s možností krácení cyklů, na němž jsou navázány dvě nezávislé jednotky srovnání/zachycení (compare/capture). Volně běžící čítač má přeřazenou programovatelnou děličku se 7 dělícími poměry v násobku 2 (2,4,8,16,32,64). Osmou možností je čítání externího kmitočtu přivedeného na pin PTA2.

Každá z obou jednotek compare/capture je schopna zachytit do 16-bitového registru stav čítače při výskytu nástupné/sestupné hrany na řídícím vstupu, s možností přerušení. Další funkcí je porovnání uložené 16-bitové hodnoty s hodnotou volně běžícího čítače. Při shodě je pak příslušný výstupní pin vynulován, nastaven nebo invertován, zase s možností přerušení. Poslední funkcí je generování PWM signálu.

4.7 Vývojové kity a software

Pro návrh a realizaci aplikací určené provozu na architektuře mikrokontrolérů řady HC08 existuje celá řada vývojových nástrojů a podpůrných obvodů s různou úrovní ladění a testování. Pro návrh, simulaci a testování zdrojových kódů jsou k dispozici následující software:

- **P&E Micro –ICS08**: Obsahuje kompletní vývojové prostředí assembleru pro celou rodinu HC08. Má zabudovaný programátor, simulátor, obvodový simulátor a debbuger.
- **Metrowerks – CodeWarrior for HC08**: Vývojové prostředí CodeWarrior určené pro HC08 umožňuje psát jak assemblerovské programy, tak programy v jazyce C/C++. Samozřejmostí je programátor a debbuger spolu s dalšími vývojovými nástroji.
- **Cosmic – IDEA08**: Kompletní prostředí pro vývoj aplikací v jazyce C s jednodušším přístupem pro začínající programátory. Verze IDEA08 je dispozici volně ke stažení pro velikost výstupního kódu do 4kB.

Hardware, pro samotné programování mikrokontrolérů řady HC08 a hlavně samotné testovaní a ladění, zastupují především tyto vývojové kity:

- **Kit JANUS**: Je koncipován jako velmi jednoduchý a snadno dostupný nástroj pro demonstraci možností mikrokontroléru, pro návrh i malosériové programování. Dostupný jako finální výrobek nebo stavebnice vybavený servisními konektory (MON08, MINIMON). Deska podporuje mikroprocesory 68HC908QT nebo 68HC908QY.
- **Kit LJ12EVB**: Vývojová deska přímo osazená mikrokontrolérem 68HC908LJ12CFU s řadou demonstračních periférií vhodná pro studijní účely. Umožňuje přímo ladit aplikace využívající LCD displej, 14 tlačítek, senzor teploty, zvonek a led diod. Dovoluje použití jakéhokoli MON08 kompatibilního software (CodeWarrior, P&E Micro, atd.).

5 Ovládání robotického manipulátoru mikrokontrolérem

Na základě poznaných vlastností a možností mikrokontrolérů v podání HC08 a rozhraní běžného analogového joysticku lze nalézt způsob, jak navrhnout modul pro řízení robotického manipulátoru, v tomto případě ROB1-3, pomocí řídících signálů z joysticku.

Na jedné straně existuje poměrně známé, ale v dnešní době už méně užívané, analogové rozhraní pro připojení joysticků, zvukových a jiných zařízeních.

Na druhé straně je robotické rameno opatřené pouze jednouchými motorickými prvky ve formě servomechanismů s jednoduchým rozhraním, ale s jemným řízením, které samo o sobě není schopno činnosti.

Mezi ně vložíme a vhodně zapojíme řídící prvek, schopný jak analogového vstupu, tak i digitálního výstupu v reálném čase. V následujících kapitolách jsou ukázána východiska pro vytvoření takového modulu jak pro samostatnou činnost, tak po výukové a demonstrační účely.

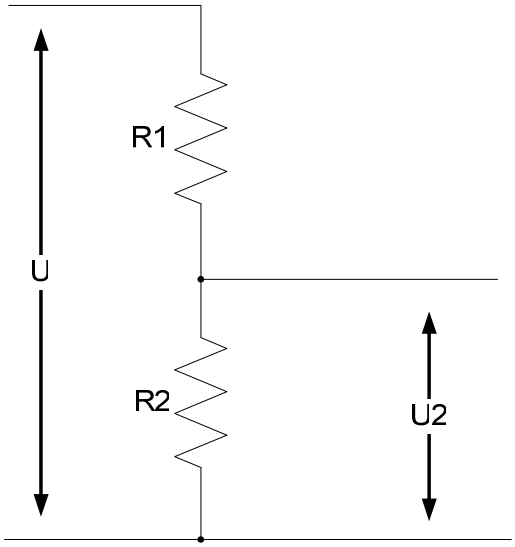
5.1 Analýza modulu ovládání robotického manipulátoru s verzí NITRON

Výběr mikrokontroléru HC08 MC68HC908QT NITRON jako řídícího a převodního prvku modulu pro ovládání robotického manipulátoru není bezdůvodný, protože demonstruje na této úloze maximální užití nejjednoduššího zástupce z HC08. Jelikož NITRON disponuje pouze 6 obecnými porty PTA, z nichž jen některé dokáží plnit i další funkce, je nutné využít každý port nejvhodnějším způsobem.

5.1.1 Analýza vstupů

Analogový joystick v základní verzi obsahuje vždy alespoň dvě nastavitelné osy X a Y a nějaké obecné tlačítko (Button 1). Zapojení joysticku, viz. *kapitola 2.1.2 a 2.1.3*, umožňuje v zásadě dva přístupy k zjistění výchylky, jednak pomoci monostabilního obvodu, který by generoval PWM pulsy nebo pomocí AD převodu s určitým rozlišením. Použití monostabilního obvodu je více časté, protože je výstupní signál lépe zpracovatelný. Přesto monostabilní obvod představuje další obvodovou součástku, kterou je možno, při využití

integrovaného AD převodníku na mikrokontroléru, vypustit. Aby bylo možno hodnotu na mikrokontroléru načíst, je třeba zapojit jednotlivé potenciometry joysticku do obvodu děliče odporu viz. *obrázek 5.1.1.1*, kde U_2 představuje měřenou hodnotu napětí.



$$U_2 = \frac{R_2}{R_1 + R_2} U$$

Obrázek 5.1.1.1: Princip odporového děliče

Po připojení napájecího napětí se na dělícím odporu objeví napětí v rozmezí 1.6V až 5V dle Ohmova zákona. Jednotlivá tlačítka připojují vývody na zem a je tedy nutné vstupy na mikrokontroléru opatřit Pull-up rezistory. Ke zjištění velikosti napětí je NITRON vybaven čtyřmi vstupy AD0, AD1, AD2 a AD3 s 8 bitovým rozlišení, které by byly schopny navzorkovat vstupní napětí a dále ho pak v mikrokontroléru zpracovat. Jelikož s použitím více tlačítek by na mikrokontroléru chyběly důležité vývody pro obsluhu servomechanizmů, stačilo by využít pouze jediného tlačítka, které by přepínalo funkci z některých už využitých os joysticku. Tlačítko může být připojeno na jakýkoli zbylý vývod.

5.1.2 Analýza výstupů

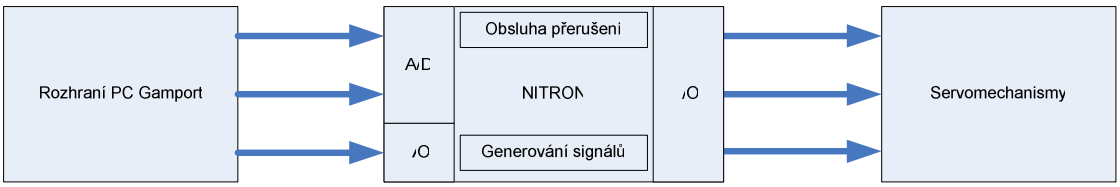
Signál pro ovládání natočení servomechanizmů, tak jak je popsán v *kapitole 3.1*, vyžaduje generování periodického obdélníkového signálu s malým kmitočtem 50Hz. Pro obsluhu časovaní má NITRON vestavěný dvoukanálový časovací obvod. Přestože tento časovací obvod dokáže měnit hodnoty na výstupech TCH0 a TCH1, je robotické rameno vybaveno třemi servomechanizmy a bylo by tak nutné třetí signál generovat později v programu smyčkou v programu. Jiným řešením je využít možnost generování přerušení na události

přetečení čítače tak, že čítá se stejnou frekvencí kroku, ale různě dlouhými periodami událostí přerušení, která budou jednak měnit hodnoty výstupů , jednak nastavovat hodnoty přetečení.

Celkem by rozhraní joysticku zabralo 3 vstupy a rozhraní servomechanismů 3 výstupy, tedy mikrokontrolér by byl plně vytížen a představoval tak z hlediska výběru optimální řešení. Případné napěťové nesrovnalosti by řešily další obvodové součástky.

5.2 Návrh modulu ovládání robotického manipulátoru s verzí NITRON

Jelikož klíčovým prvkem modulu je mikrokontrolér HC08 NITRON, je celkový návrh modulu přizpůsoben fyzické realizaci tohoto mikrokontroléru v provedeni DIL8, z pohledu implementace je navíc uvažován vnější zdroj stabilizovaného napětí a bude také brán ohled na velikost celého řešení tak, aby výsledný modul byl vhodný k běžnému použití.

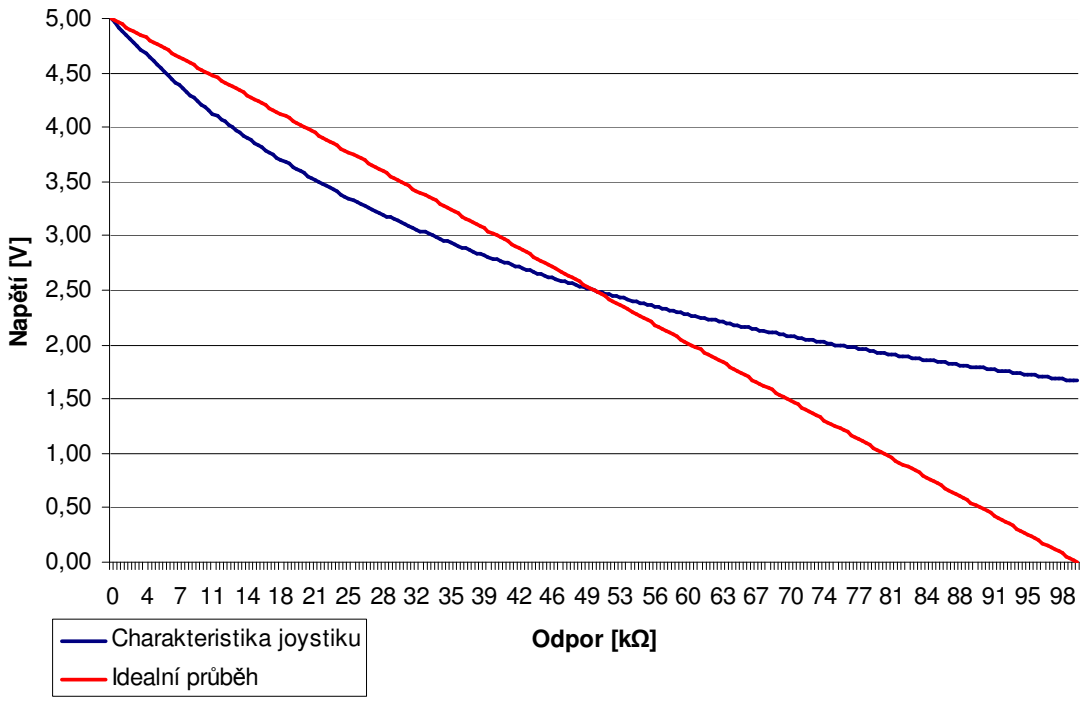


Obrázek 5.2.1: Blokové schéma s mikrokontrolérem NITRON

Na obrázku *5.2.1* je blokové schéma zapojení modulu pro ovládání robotického manipulátoru s použitím mikrokontroléru HC08 NITRON, jehož podrobné schéma je v *příloze 1*.

5.2.1 Rozhraní PC Gameport

Blok modulu reprezentovaný konektorem D-SUB-F s 15 vývody, z něhož jsou použity pouze vývody Joystick 1- X (3. vývod), Joystick 1 – Y (6. vývod), B1 (2.vývod) a vstupy +5V (1. vývod) a uzemněné vývody (4. a 5. vývod). Analogové vývody 1 - X a 1 – Y jsou uzemněny přes dělící odpory 50kΩ. Napětí na těchto odporech se mění dle rovnice viz. *obrázek 5.1.1.1* v rozsahu 1.6V až 5V se středem v 2.5V. Jelikož tento průběh není lineární, jak ukazuje *obrázek 5.2.1.1*, bylo nutné vzniklé odchylky odstranit převodní tabulkou v programu mikrokontroléru, aby změna přírůstku napětí byla co nejvíce lineární. Analogový signál je veden k portům PTA0 a PTA1, na port PTA2 s povoleným pull-up rezistorem je připojeno tlačítko B1. Pro rozhraní se předpokládá použití standardního joysticku.



Obrázek 5.2.1.1: Graf závislosti měřeného napětí na odporu joysticku

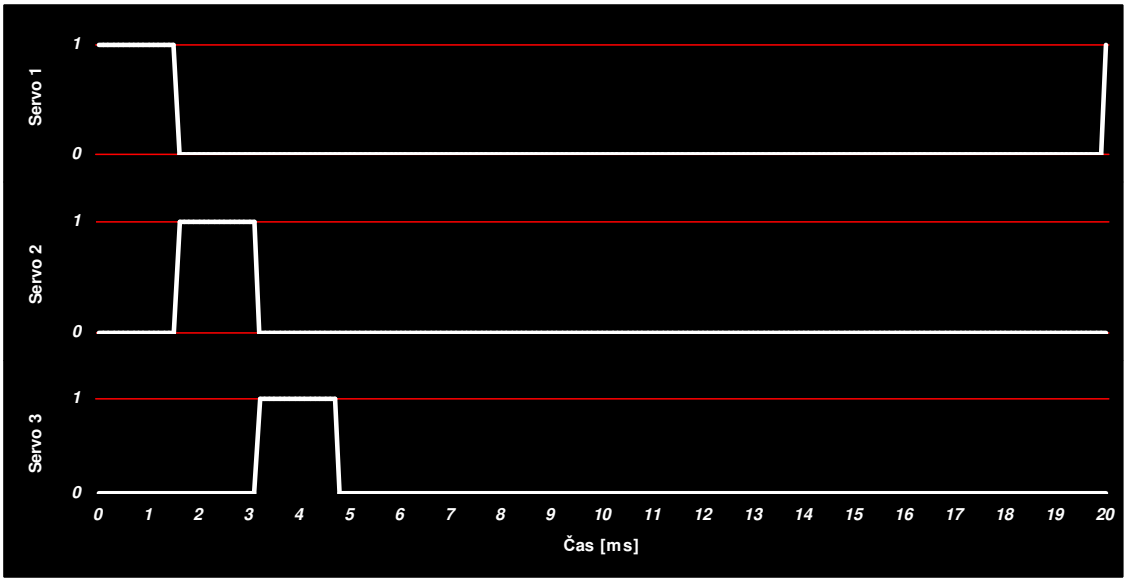
5.2.2 Mikrokontrolér NITRON

Na základě vstupních hodnot z A/D převodníků a vstupu tlačítka vypočítává délku výstupních pulsů PWM modulace na rozhranní serv manipulátoru. Zpracovává události přetečení časovače, dokončení AD převodu a obsluhy vstupu tlačítka. Implementace programu pro mikrokontrolér je v kapitole návrhu programu. Z pohledu zapojení na desce, je zohledněno zajištění stabilního napájecího napětí mimo napájení servomechanismů, tak aby činnost serv nezapříčinila ztrátu napětí a nekorektní funkci mikrokontroléru.

5.2.3 Rozhraní servomechanismů

Vychází z kapitoly *3.1*, je připojeno na konektor 3x3 pin, kam je přivedeno jednak napajení a jednak výstupní signály z mikrokontroléru. Obslužný program mikrokontroléru zajišťuje generování půlsů pro jednotlivé servo mechanismy na portech PTA3-5 s periodou 20ms a šířkou pulsu 0.9ms – 2.1ms. Podle tvaru řídícího signálu nastavují úhel natočení svých os v rozsahu až ± 54°. Tento výstup nebylo nutné nijak posilovat, avšak je třeba zajistit dostatečný zdroj proudu a jeho odrušení, tak aby proudový odběr otačejících se serv nezpůsoboval pokles napětí na

mikrokontroléru. S použitými servy HS-322 byl změřen odpěr max.600mA. Na *obrázku 5.2.3.1* je zobrazen průběh výstupního signálu na jednotlivých servech.



Obrázek 5.2.3.1: Časový průběh generování výstupních hodnot pro servomechanismy

5.3 Ovládací program pro mikrokontrolér NITRON

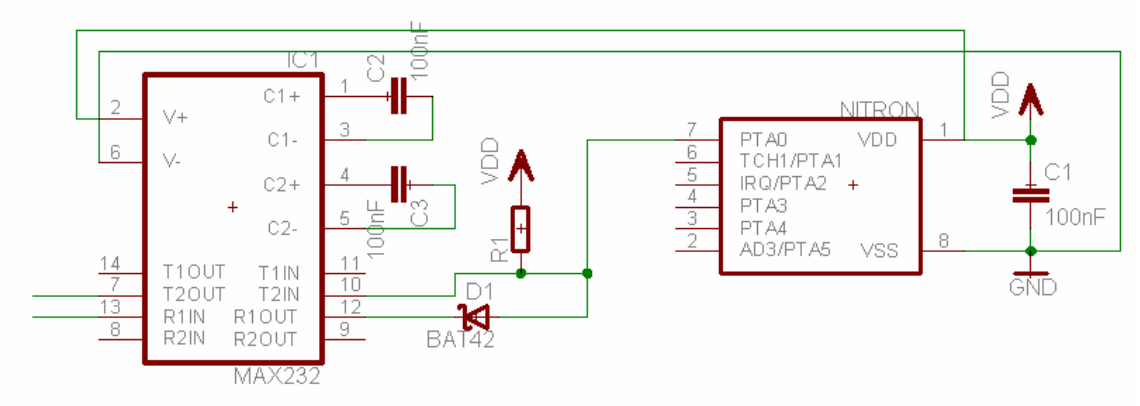
5.3.1 Programování mikrokontroléru

Pro realizaci příslušné funkce mikrokontroléru je potřeba vytvořit, přeložit a nahrát instrukční soubor programu do mikrokontroléru. Jak bylo zmíněno v kapitole 4.7, existuje řada nástrojů poskytujících více či méně sofistikovaný přístup k jednotlivým fázím programování. Pro potřeby této práce byl zvolen nástroj firmy Metrowerks, především kvůli velmi silnému vývojovému prostředí, které lze pro velikosti kódů do 4KB získat zdarma a také vzhledem přehlednému ladícímu a simulačnímu nástroji.

Pro implementace byl zvolen programovací jazyk C, protože jsou v něm jednotlivé části kódu přehledné a dobře demonstruje nasazení vyšších programovacích jazyků i pro malé kódy. Efektivitu kódu psaného v jazyku C lze ohodnotit nejen velikostí výstupného kódu, ale i přímo překladem do symbolických instrukcí.

Pro naprogramovaní mikrokontoléru NITRON nebyl přímo využit vývojový Kit Janus, ale jeho zjednodušená verze bez MINIMON tlačítka, resetu, přidaných diod a tlačítek, tak aby naprogramováni proběhlo s co nejmenším počtem vodičů na nepájivém poli a demonstrovalo

tak jednoduchost vytvoření zázemí pro programovaní mikrokontrolérů obecně. Programování mikrokontroléru LJ12 probíhá už přímo na kitu.



Obrázek 5.3.1.1: Schéma programátoru pro mikrokontrolér NITRON

5.3.2 Analýza programu pro mikrokontrolér

Na základě návrhu modulu ovládání robotického manipulátoru z *kapitoli 5.2* a funkčních možností mikrokontroléru NITRON, především jeho jednotlivých bloků, je nutné v ovládácím programu navrhnut jednak zpracovaní vstupních parametrů z joysticku (časově a výpočetně náročnější) a jednak generování výstupních signálů (citlivé na přesnost).

Blok AD převodníku, který by měl digitalizovat hodnotu napětí na portech PTA0 a PTA1, ukládá hodnoty v rozsahu jednoho bajtu tak, že pro hodnotu blízkou zápornému pólu zdroje ukládá 0 a pro hodnotu blízkou kladnému pólu zdroje hodnotu 255. Vstupní rozsah napětí na AD převodníku přes dělič napětí je v rozsahu 1.6V až 5V nelineárně, což odpovídá hodnotám 84 až 255. První možností, jak s těmito hodnotami pracovat, je zanedbání nelinearity a omezeného rozsahu napětí, což by ale ve výsledku znamenalo výrazně neekvivalentní výchylky v ovládání. Další možností je nalezení korigující funkce, která by lépe kopírovala charakteristiku napětí, ale na druhé straně by vyžadovala nemalý čas k svému výpočtu. Poměrně slušného výsledku je možné docílit s použitím přepočítané korekční tabulky hodnot tak, aby pro získanou hodnotu napětí existovala hodnota v tabulce, která by byla dále zpracovávána. Nevýhodou těchto korečních způsobů je silná závislost na charakteristice kokrétního typu ovládaícho joysticku.

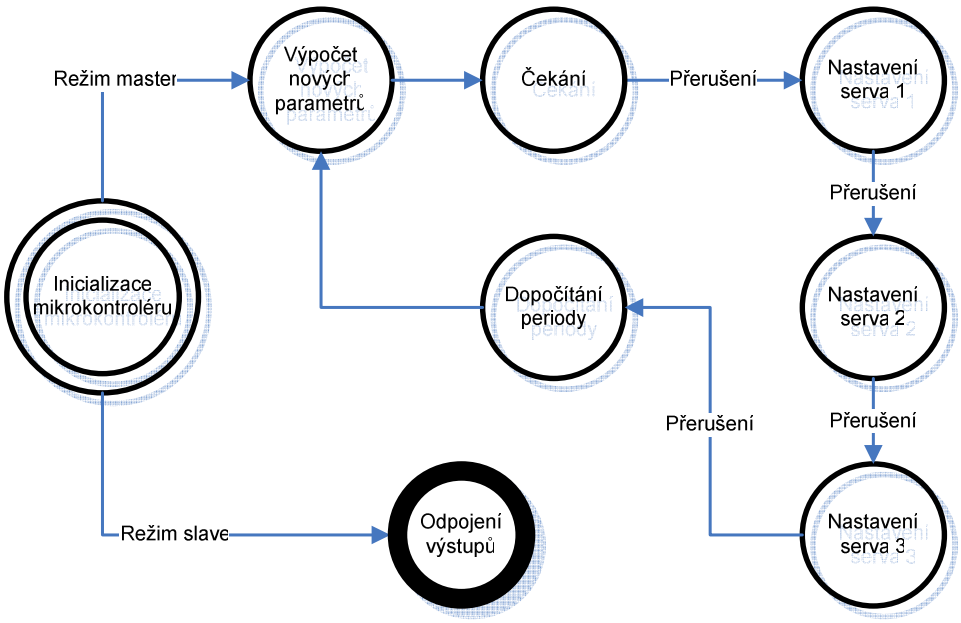
Jelikož blok časovače obsahuje pouze dva nezavislé kanály, musel by být výstup na třetí servo ovládán nějakou programovou smyčkou s velkou časovou nejistotou, která by mohla způsobovat neplynulý pohyb ramen robota. Pokud by časovací blok pouze generoval přerušení při svém přetečení a obsluha toho přerušení by zajistila změnu logických úrovní na

výstupech PTA3-5, tak by chyba v periodě představovala jen dobu několika instrukcí v obsluze přerušení.

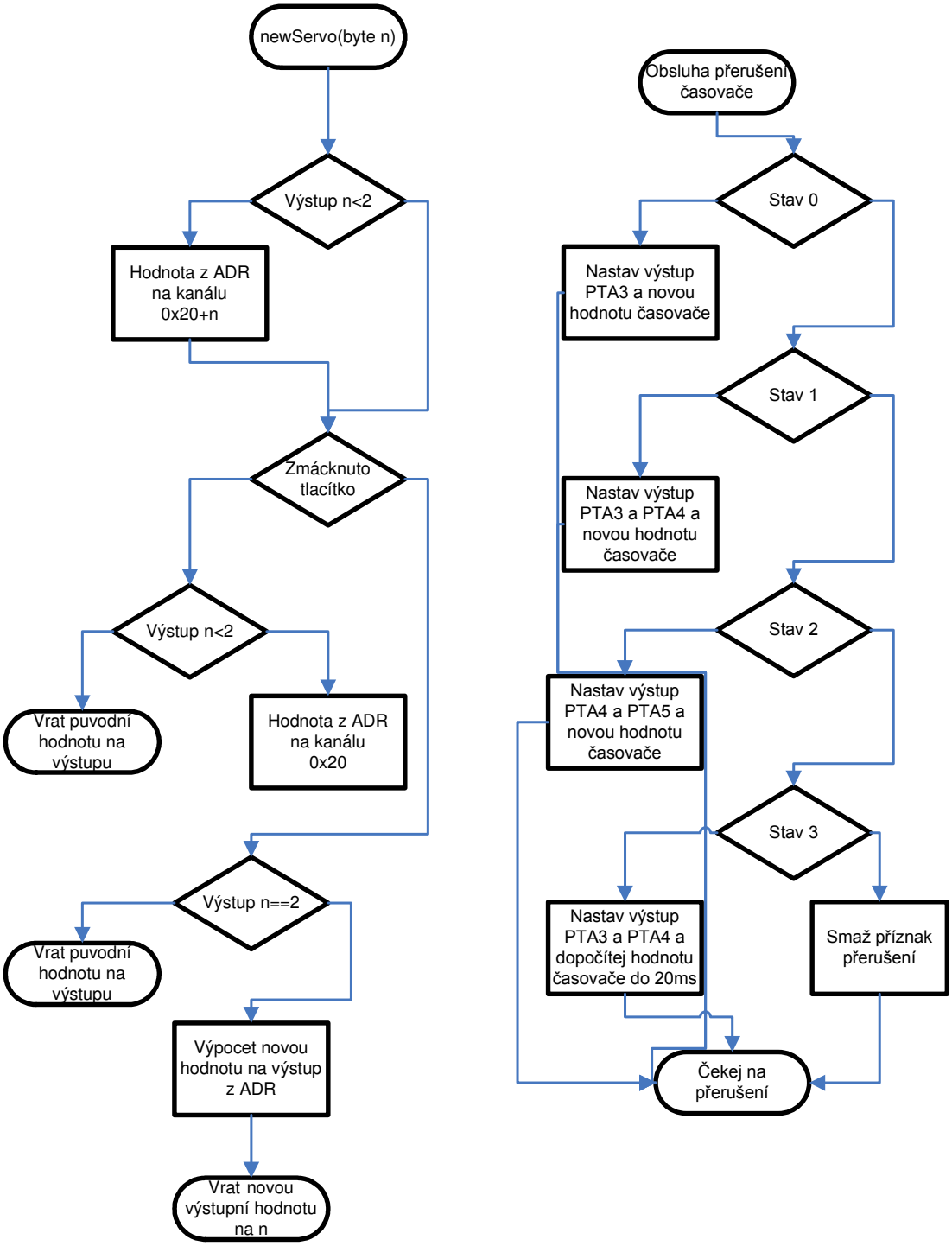
Zpracování údajů z registrů bloků časovače a převodníku a nastavení hodnot nových pro výstupy může představovat delší časový úsek, který by měl být prováděn mimo oblast generování výstupních signálu. Bez vytvoření určitého stavového automatu nad chováním mikrokontroléru, by obsluha vstupů výrazně ovlivňovala činnost výstupů. Vytvoření modelu chování na úrovní vytváření více obsluh přerušení, by sice snížilo riziko kolizí, ale nevyřešilo stabilitu period výstupních signálů.

5.3.3 Návrh programu pro mikrokontrolér

Z návrhů řešení vyplynul následující stavový automat chovaní mikrokontroléru (viz. *obrázek. 5.3.3.1*) a vývojové diagramy význačných úseků program (viz. *obrázek 5.3.3.2*).



Obrázek 5.3.3.1: Stavový automat řízení pro mikrokontrolér NITRON



Obrázek 5.3.3.2: Vývojové diagramy funkcí newServo a obsluhy přerušení časovače

Funkce a procedury

Jméno	Vstup	Výstup	Popis
initServos	void	void	Inicizialize hodnot pro servomechanismy.
initTimer	void	void	Inicializace konfiguračních registrů časovače pro takt 800kHz.
correctAD	byte input	byte	Vrácí korigovanou hodnotu z **input** podle hodnot v korekční tabulce.
newServo	byte n	short	Vypočítá novou hodnotu taktů časovače pro konkrétní servo **n** z příslušného AD převodníku a poslední hodnoty serva.
setServos	void	void	Nastaví nové hodnoty počtu taktů pro všechna serva.
isSlave	void	byte	Funkce pro mikrokontrolér NITRON k zjištění, zda je mikrokontrolér po startu připojen k nadřazenému mikrokontroléru LJ12.
setDelay	short delay	void	Resetování časovače a nastavení nové čítací hodnoty, dle parametru **delay**.
TIMint	void	void	Obsluha přerušení přetečení časovače a nastavení logických úrovní na příslušných výstupech. Implementuje stavový automat.
Main	void	void	Hlavní program v nekonečné smyčce se stavovým řízením.

Tabulka 5.3.3.1: Popis funkcí a procedur

5.3.4 Linearizace odporové charakteristiky joystiku

Zapojení joysticku v obvodu děliče napětí má za následek napěťovou charakteristiku, viz. *obrázek 5.2.1.1*. Přímou degradací této křivky na přímku má za následek zpracování výrazně odlišných hodnot oproti skutečné výchylce. Eliminace této chyby závisí na schopnosti vypočítat korekční hodnotu pro každou možnou pozici joystiku. Nelezení funkce a následné dopočítání její hodnoty v bodě představuje zbytečné vytížení mikrokontroléru, který získanou odchylku stejně degraduje v procesu přepočtu. Skutečným faktorem hodnotícím správné vychýlení je schopnost operátora a mechanickým provedením joystiku. Pro nalezení odchylky

byla využita metoda převodní tabulky, která pro aktuální hodnotu napětí vrací její lineární ekvivalent, což vyřeší nelineární řízení v jednotlivých úsecích pracovního prostoru joystiku a zároveň neklade přílišné nároky na výpočetní výkon mikrokontroléru na úkor paměťových nároků. Na následující tabulce jsou zobrazeny vypočtené hodnoty odchylek se čtyřnásobnou hrubostí, tak jak je uvozována při odpočtu odchylky v NITRONU.

index	diff_index	index	diff_index
0	**-84**	22	**22**
1	**-72**	23	**22**
2	**-60**	24	**22**
3	**-50**	25	**22**
4	**-40**	26	**22**
5	**-32**	27	**21**
6	**-25**	28	**21**
7	**-18**	29	**20**
8	**-12**	30	**19**
9	**-7**	31	**18**
10	**-3**	32	**17**
11	**0**	33	**16**
12	**5**	34	**15**
13	**8**	35	**13**
14	**11**	36	**12**
15	**13**	37	**10**
16	**15**	38	**9**
17	**17**	39	**7**
18	**18**	40	**5**
19	**20**	41	**3**
20	**20**	42	**1**
21	**21**		

Tabulka 5.3.4.1: Tabulka přepočítaných rozdílových indexů převodního pole

Pokud uvažujeme pole hodnot na 1Byte, které obsahuje všechny hodnoty v rozsahu 0-255 tak, jak je například vrací registr AD převodníku, pak je výsledná hodnota, kterou by vrátila převodní funkce vypočítána následujícím způsobem:

$$new\_value = old\_value + array[old\_value - 21]$$

Konstanta 21 představuje rozdíl 0V a 1.6V tak, aby se uložila pouze užitečná část rozsahu, zbytek není uvažován.

5.3.5 Specifikace testů

1. **Přepnutí do režimu slave**

   Pokud se objeví hodnota napětí rovna log. 1 na vývod PTA5 mikrokontroléru v čase jedné periody od zapnutí napájení, tak přechází mikrokontrolér do režimu slave, ve kterém jsou odpojeny veškeré výstupy tak, aby neovlinil nadřízený prvek. V tomto režimu je mikrokontrolér neustále resetován vlivem povoleného obvodu COP, což po odpojení napětí na PTA5 umožní spustit kód porgramu pro regulérní řídící činnost. Z tohoto stavu už nelze mikrokontrolér přepnout, ale je nutno odpojit od zdroje napětí.

2. **Trvalé vychýlení jedné čí více os joysticku**

   Vychýlením osy joysticku do jedné z krajních pozic dochází v mikrokontroléru k inkrementaci/dekrementaci šířky pulsu na výstupu a změně pozice ramena robotického manipulátoru. Tento proces pokračuje do doby, než je dosažena programová ochranná hranice, kdy už není šířka pulsu snižována/zvyšována a výchylka ovladače už není započítávána.

3. **Snížení napájecího napětí**

   Snížením napájecího napětí nebo použitím slabého zdroje dochází k nekorektnímu provozu mikrokontroléru, která je patrná zvláště při nastavování výchylek servomechanismů robotického ramena. Připojením stabilizovaného zdroje napětí je regulérní činnost obnovena.

5.3.6 Vlastní implementace

Program ovládání robotického manipulátoru pro mikrokontrolér NITRON se skládá z dvou hlavních částí. První část obsahuje funkce pro sběr dat z AD převodníků, tlačítka a zpracování

jejich hodnot pro výpočet aktualní doby jednotlivých pulsů na výstupech asynchronně. Druhá část představuje stavový automat v časovacím bloku, pro spravné načasování změn logických úrovní na výstupních portech.

Po připojení napájecího napětí na vývody VSS a VDD se začne provádět kód hlavního programu, v němž je nejprve ověřeno, zda není mikrokontrolér na základě logické úrovně. 1 na vstupu PTA5, zapojen v režimu SLAVE. V tomto režimu se předpokládá, že mikrokontrolér je na stejné sběrnici s jiným ovládacím prvkem např. LJ12, který přebírá řízení servomechanismů a mikrokontrolér odpojí své výstupy. V opačném případě jsou postupně inicializovány pomocné výpočetní proměnné, registry časovače, AD převodníku a IO porty. Frekvence časovače je nastavena v poměru 1:4 a čítač se tak inkrementuje s periodou 1,25us. Tento poměr je zvolen vzhledem 16b rozlišení časovače a generovaní přerušení i po 20ms. V prvním stavu automatu jsou vypočítaný hodnoty serv pro časovač, tento proces je asynchronní a probíhá pouze jednou v ramci periody signálu serv. Vícekanálový AD převodník je postupně nastavován na vstupy PTA0 a PTA1, získané hodnoty v rozsahu <0,255> jsou nejprve korigovány přes pomocnou tabulku předpočítaných hodnot a následně upraveny do rozsahu <-6,6>, z něhož jsou jednotlivé hodnoty použity pro změnu pulsu serva v rámci jedné periody. V nasledujících stavech je vždy změněna hodnota na jednom, či dvou portech současně a nastavena nová doba čítaní časovacího obvodu tak, aby pulsy na jednotlivá serva byla generována postupně za sebou. V posledním kroku je dopočítán zbývající čas do periody 20ms a nastaven stav automatu do stavu výpočtu nových hodnot jednotlivých serv pro další periodu. Činnost mikrokontroléru už není nijak přerušitelná, v systémových registrech mikrokontroléru je vypnut hlídací obvod pro reset, ale vzhledem k reakcí systému na jediné přerušení nenastalo v běžném provozu jeho zacyklení. Použitý vnitřní oscilátor na 3,2MHz nevykazoval žadné větší změny frekvence a nebylo třeba ho zvláště upravit.

5.3.7 Zdrojové kódy

- Zdrojový kód programu pro mikrokontrolér NITRON se nachází *v příloze 3*.

5.4 Analýza modulu ovládání robotického manipulátoru pro Kit LJ12EVB

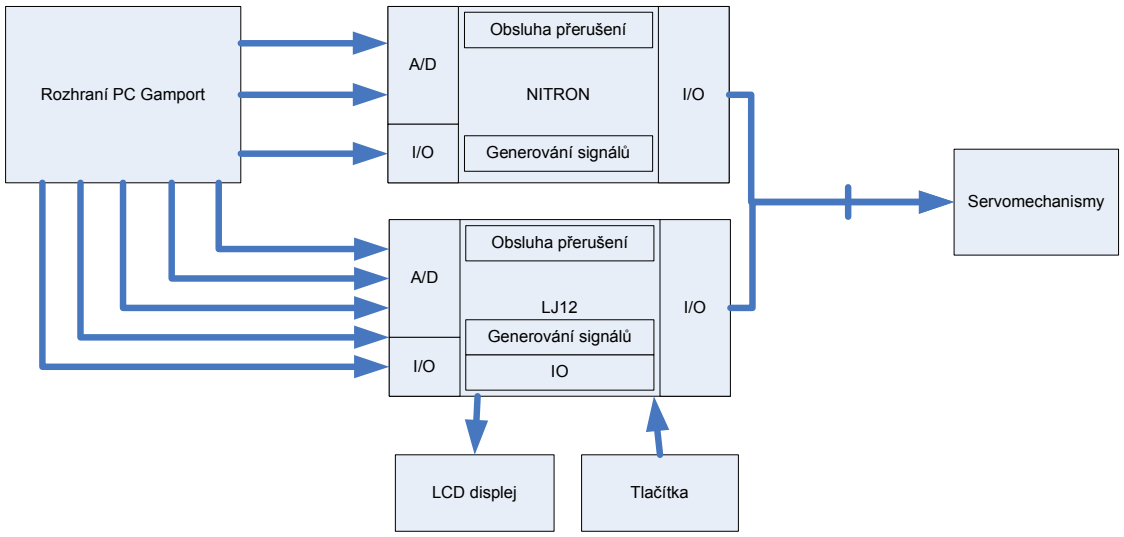
Pro účely demonstrace a dalšího případného rozšiřování aplikací pro ovládání robotického manipulátoru i na laboratorních přípravcích zastoupených kitem LJ12EVB je nutno předešlý návrh upravit, a to jedním z následujících způsobů. První možností je navržení nové převodní desky, především pro jednodušší zapojení konektorů joysticku a jednotlivých serv a následné připojení k vývojovému kitu. Druhou možností je využití předešlého návrhu k jeho rozšíření o další konektory. Jelikož téma práce nekoliduje s použitím druhé kombinované možnosti, jsou dále uvažovány různé přístupy k využití portů mikrokontroléru vývojové desky ve vztahu návrhu k existujícímu řešení.

Kit LJ12EVB, jak vyplývá s označení, obsahuje integrovaný, výše popsaný mikrokontrolér HC908LJ12, který disponuje 32 obecnými porty, z nichž některé jsou už napevno propojeny s dalšími komponenty, např. LCD řadič, tlačítka, sériový port atd. viz *příloha 5*. Ze všech těchto vývodů jsou návrháři k dispozici pouze vývody na konektoru X2 a X3 s tím, že vývody na X3 slouží především jako MINIMON ladící rozhraní. Konektor X2, který asi nejlépe vyhovuje jako vstup/výstupní konektor pro zapojení externího modulu, obsahuje 16 vývodů z nichž jsou 4 napájecí, 2 sériové a zbylých 12 datových vstupně/výstupních linek. Na rozdíl od minimalistické verze s NITRONem se otvírá možnost využití více výstupu analogového joysticku, možnost sledování více os nebo více tlačítek. Jelikož má mikrokontrolér LJ12 jeden časovač navíc, mohou být signály pro jednotlivá serva obsluhována nezávisle. Předností vývojového kitu je možnost rozšíření o dynamickou kalibraci rozsahů pomocí integrovaných tlačítek a zobrazování výchylky na zabudovaném displeji.

Do řešení s použitím pouze samotného NITRONu je nutné přidat další konektor pro vstupy a výstupy a zajistit elektrické oddělení výstupních signálů. Je možné uvažovat i oddělení signálů pomocí dalších logických členů, které by představovaly další prostor navíc.

5.5 Návrh modulu ovládání robotického manipulátoru pro kit LJ12EVB

Pří návrhu zapojení se vychází z návrhu jednodušší verze pouze s NITRONem, který je rozšířen o další konektor pro rozšiřující připojení k LJ12EVB na konektor X2 i s možností napájení. Přestože vstupní hodnoty zpracovávají oba mikrokontroléry, výstup je oddělen tak, aby pouze jeden mikrokontrolér ovládal serva robotického manipulátoru.



Obrázek 5.5.1: Blokové schéma s mikrokontrolérem NITRON a LJ12

Na *obrázku 5.5.1* je blokové schéma zapojení modulu pro ovládání robotického manipulátoru s použitím kombinovaného přístupu nezávislého mikrokontroléru HC08 NITRON a současně rozšíření pro připojení k laboratornímu kitu LJ12EVB, jehož podrobné schéma je v *příloze 2*.

5.5.1 Mikrokontrolér LJ12

V zapojení kitu LJ12EVB se zapojenými vývody (5., 6., 7., 8., 9., 10., 12. a 14.) na konektoru X2. Vstup je rozšířen o další vývod pro případnou hodnotu třetího analogového výstupu z joysticku a dalšího tlačítka. Počet výstupů je stejný jako pro NITRON s tím, že jednotlivé vývody prochází zpět na společné vývody s NITRONem, který je odpojen přes logickou úroveň 1 na portu PTA5.

5.5.2 LCD displej

Kit LJ12EVB je vybaven sedmi 7-segmentovými prvky, které je schopen ovládat přes speciálních 25 vývodů. Displej slouží k zobrazení výchylek servomechanismů v rozsahu 0-99 v pořadí os x, y a z.

5.5.3 Tlačítka

Vestavěný blok klávesnice připojený na porty PTD a PTC slouží k pro demonstraci záznamu až osmi pozic do paměti a následnému postupnému průchodu jednotlivými uloženými pozicemi. Obsluhu tlačítek zajišťují porty PTD4-PTD7 a PTC4-PTC6.

5.6 Ovládací program pro mikrokontrolér LJ12

5.6.1 Analýza programu pro mikrokontrolér LJ12

Pro analýzu návrhu lze s výhodou použít již navržené schéma ovládání s verzí NITRON a zachovat elektrické zapojení a rozšířit ovládání o nové prvky a periferie. Navíc mikrokontrolér LJ12 disponuje novými a lepšími parametry, které mohou zlepšit již existující návrh.

Pro AD převod je vhodné, z hlediska propojení se samostatným modulem, opět použít vstupy portu PTA, které dokáží pracovat až s 10b rozlišením, které by případně mohlo zvýšit přesnost a jemnost ovládání servomechanismů. Vstupní rozsah napětí na AD převodníku zůstává přes dělič napětí v rozsahu 1.6V až 5V nelineárně. Problém odstranění nelinearity odporu joysticku byl s verzí NITRON řešen korekční tabulkou s 43 prvky z důvodu omezené paměti. Citlivost řízení u NITRONu nebyla velmi přesná. Mikrokontrolér LJ12 disponuje větším paměťovým prostorem a nabízí tak zvýšení přesnosti ovládání.

Stejně jako u mikrokontroléru NITRON je možné i zde výstupní signály generovat s použitím přetečení časovače, navíc ale LJ12 dovoluje nasazení už dvou nezávislých časovačů a generování signálu tedy může plně zastoupit PWM modulace na jednotlivých kanálech. V zapojení kitu LJ12EVB nejsou tyto výstupy připojeny na konektor X2 a nelze je tedy v tomto návrhu použít.

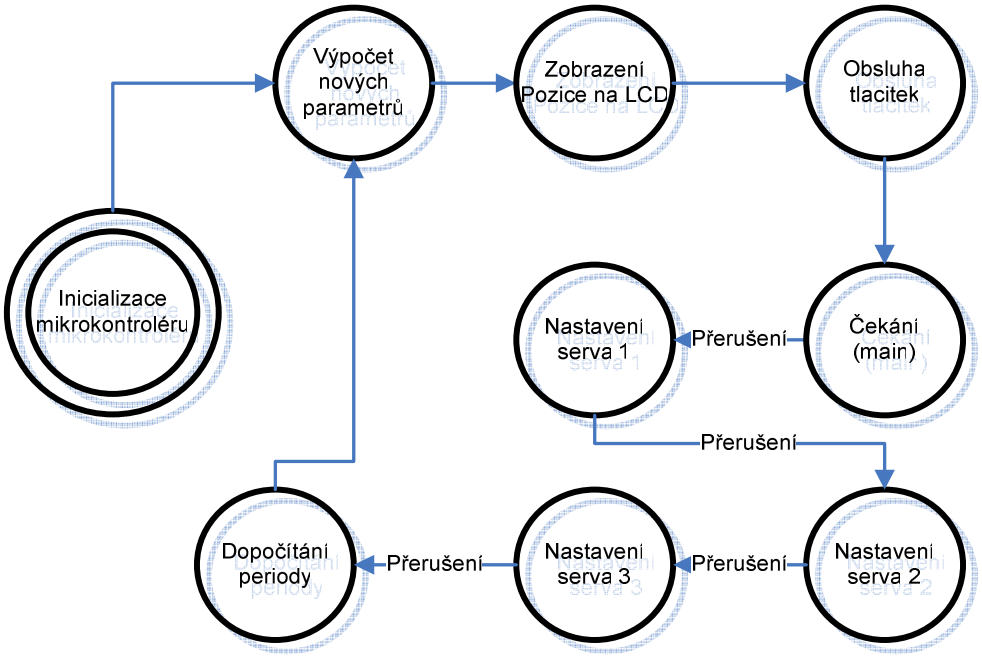
Pro vizuální kontrolu aktuální pozice servomechanismů a vytvoření tak určitého definovaného pracovního prostoru robotického manipulátoru by měl posloužit integrovaný LCD displej, jehož řízení zajišťuje integrovaný řadič v mikrokontroléru LJ12. Způsob zobrazení informace na displeji je dána jeho parametry a dovoluje zobrazení max. třech dvou-

ciferných čísel, což by dovolilo zobrazit hodnoty o polohách všech tří serv. Nevýhodou je skutečnost, že vzorkování polohy servomechanismy je jemnější a rameno se nemusí nacházet pro stejnou hodnotu na stejném místě. Tento problém by musel být řešen postupným zobrazováním jednotlivých parametrů dalšími ovládacími prvky.

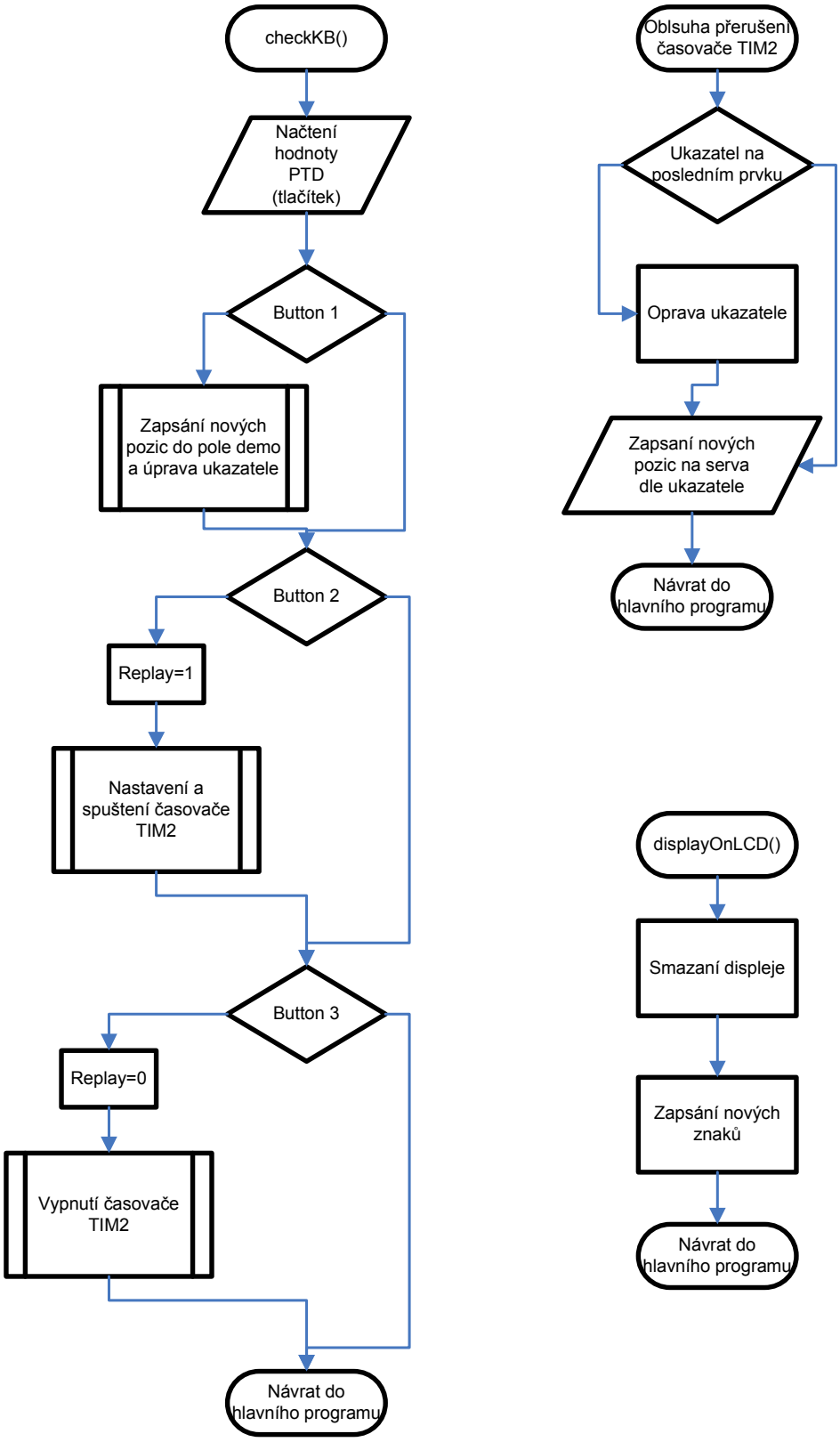
Blok klávesnice demonstračního kitu může výrazně rozšířit možnosti konfigurace parametrů ovládání, zvláště pokud je používáno více různých joysticků. Pro demonstrační učely by bylo vhodné, aby bylo možno ukládat jednotlivé pozice, do kterých se rameno dostalo a následně s určitým časovým zpožděním tyto nahrané pozice postupně procházet. K ovládání toho makro režimu by byla využita právě tlačítka klávesnice s danými funkcemi.

5.6.2 Návrh programu pro mikrokontrolér LJ12

Uvažovaná realizace ovládacího programu pro mikrokontrolér LJ12 zobrazená na *obrázku 5.6.2.1.* vychází z realizace na mikrokontroléru NITRON s přidáním nových stavů řízení a zobrazení hodnot. Navržená část pro demonstraci předdefinovaného chování je přiblížena ve vývojovém diagramu viz. obrázek *5.6.2.2.*



Obrázek 5.6.2.1: Stavový automat řízení pro mikrokontrolér LJ12



Obrázek 5.6.2.2: Vývojové diagramy funkci checkKB a obsluhy přerušení časovače TIM2

Funkce a procedury

Jméno	Vstup	Výstup	Popis
initServos	void	void	Inicizializuje hodnot pro servomechanismy.
initTimer	void	void	Inicializace konfiguračních registrů časovače pro takt 800kHz.
correctAD	byte input	byte	Vrácí korigovanou hodnotu z **input** podle hodnot v korekční tabulce.
newServo	byte n	short	Vypočítá novou hodnotu počtu taktů časovače pro konkrétní servo **n** z příslušného AD převodníku a poslední hodnoty serva.
displayOnLCD	void	void	Zapsání aktuální hodnoty výchylky serva na integrovaný LCD displej ve fortmatu xx:yy:zz.
checkKB	void	void	Obsluha stisku tlačítek 1,5,7 a následný zápis nové pozice, průchod pozicemi nebo ukončení průchodu pozicemi.
setServos	void	void	Nastaví nové hodnoty počtu taktů pro všechna serva.
setDelay	short delay	void	Resetovaní časovače a nastavení nové čítací hodnoty, dle parametru **delay**.
TIMint	void	void	Obsluha přerušení přetečení časovače TIM1 a nastavení logický úrovní na příslušných výstupech. Implementuje stavový automat.
DEMOint	void	void	Obsluha přerušení od časovače TIM2 pro postupné nastavení hodnot servomechanismů z pole uložených hodnot v cca 2s intervalech.
Main	void	void	Hlavní program v nekonečné smyčce s stavovým řízením.

Tabulka 5.6.2.1: Popis funkcí a procedur

5.6.3 Specifikace testů

- **Odpojení kitu od ovládacího modulu v průběhu činnosti**

  Modul pro ovládání robotického manipulátoru s integrovaným mikrokontrolérem NITRON je připojen přes kabel na port X2 kitu LJ12EVB. V tomto zapojení se jednotlivé prvky napájí buď přes stabilizátor na ovládací desce nebo přes zdroj na kitu LJ12EVB, který zároveň svou činností odpojuje od řízení mikrokontrolér NITRON a ovládá manipulátor. Pokud je tento kabel odpojen, dochází k přepnutí

NITRONu do master režimu a přebírá řízení serv. Odpojení napájení, pokud je vedeno přes propojovací kabel, ukončuje jakoukoli další činnost, pokud je ale modul napájen samostatně, dochází ke ztrátě několika taktů řídícího signálu a následované rychlou změnou pozice ramene.

- **Ověření činnosti tlačítek**

  Stisknutím tlačítka 1 není pozorována žádná viditelná změna chování robota, v mikrokontroléru dochází k uložení aktuální pozice do paměti. Následným stiskem 4 se rameno po cca 2s přesouvá do první uložené pozice, po následujících 2s se vlivem neuložené další pozice v paměti rameno natáčí do výchozí nulté pozice. Tento proces se opakuje v periodě 16s. Stiskem tlačítka 7 je řízení ukončeno a výchylku ramene lze ovládat změnou polohy páky joysticku.

- **Pozicování podle LCD displeje**

  S použítím joysticku je nastavena konkretní pozice ramene a jeji hodnota je odečtena na LCD displeji, následně je rameno nastaveno do jiné polohy. Postupnou malou změnou výchylky se měni pozice ramena, bez přímého pozorování ramena, ale pouze údajů na LCD displeji. Po dosažení stejných hodnot pozic je odečtena chyba, jakou se rameno vlivem hrubého rozsahu na LCD dopustí pro konkrétní hodnotu. Chyba odchylky ramene není větší než 0.1°, což vzhledem ke konstrukci robota k demonstračním účelům postačuje.

5.6.4 Vlastní implementace

Navržený program ovládání robotického manipulátoru joystickem pro laboratorní kit LJ12EVB vychází principiélně z kódu pro mikrokontrolér NITRON upraveného pro využití periferiií na kitu. Program nezlepšuje ovládání robota, ale přidává nové funkce, které mohou posloužit k laboratorním cvičením.

Připojením zdroje napětí je zaveden řídící program mikrokontroléru, který v tomto případě pro LJ12 pracuje výhradně v režimu master a neověřuje, zda generované signály mohou narušit řízení jiného prvku. Jelikož je kit LJ12EVB připojen k hodinovému signálu na kitu Janus s frekvencí 9,83MHz je frekvence sběrnice 2,4576MHz, což je méně než pro mikrokontrolér NITRON. Použití jiného taktovacího kmitočtu se odrazilo především v bloku generování výstupního sígnálu, kde je namísto předešlého poměru 1:4 použit poměr 1:2 a

čítač se tak inkrementuje s periodou 834ns. Nastavené parametry modulu registru čítače TIM1, který má na starost periodickou změnu úrovní na výstupech, jsou jednoduše násobeny koeficientem 1,5, tak aby byly zachovány parametry řízení.

Novými stavy v procesu řízení robota jsou zobrazovaní pozice servomechanismů na LCD displeji ve formátu xx:yy:zz, kde jednotlivé polohy představují aktuální vychýlení serv v rozlišení 0 až 99. Tyto hodnoty jsou zapisovány jednou za 20ms přes porty LDAT4 až LDAT12. Na každých 12μs změny šířky pulsu na vstupu serva tak připadá jedna pozice na displeji. Programová část obsluhy tlačítek a čítače TIM2 se podílí na volbě činnosti mikrokontroléru. V normálním režimu je ovládání servomechanismů přímo závislé na aktuálním stavu joystiku, ale stisknutím tlačítka 2 (na bloku klávesnice tlačítko 4) se mikrokontrolér přepíná do „automatického“ režimu, kdy v intervalu ~2s prochází postupně uložené pozice v paměti. Pro uložení pozice slouží tlačítko 1, kterým se aktuální výchylka ramene v jednotlivých osách uloží do pole 8x3B. Činnost načítání uložených pozic z tohoto pole provádí obsluha přerušení přetečení časovače TIM2 s periodou 65535, což odpovídá při dělícím poměru 1:64 době 1,75s. Ukončení čítání časovač a tedy i ukončení tohoto režimu zajistí stisk tlačítka 3 (na klávesnice tlačítko 7).

5.6.5 Zdrojové kódy

- Zdrojový kód programu pro mikrokontrolér LJ12 se nachází *v příloze 4*.

6 Statistické vyhodnocení

Program pro mikrokontrolér NITRON:

Počet řádku programu: **158**

Velikost zdrojového souboru: **3374B**

Velikost kódu a dat: **471B + 57B**

Program pro mikrokontrolér LJ12:

Počet řádku programu: **317**

Velikost zdrojového souboru: **7510B**

Velikost kódu a dat: **1063B + 109B**

Parametry ovládání:

Šířka řídícího pulsu: **900µs - 2100µs**

Max. zrychlení šířky pulsu za 1s: **375**

Min. doba průchodu celého rozsahu : **3,2s**

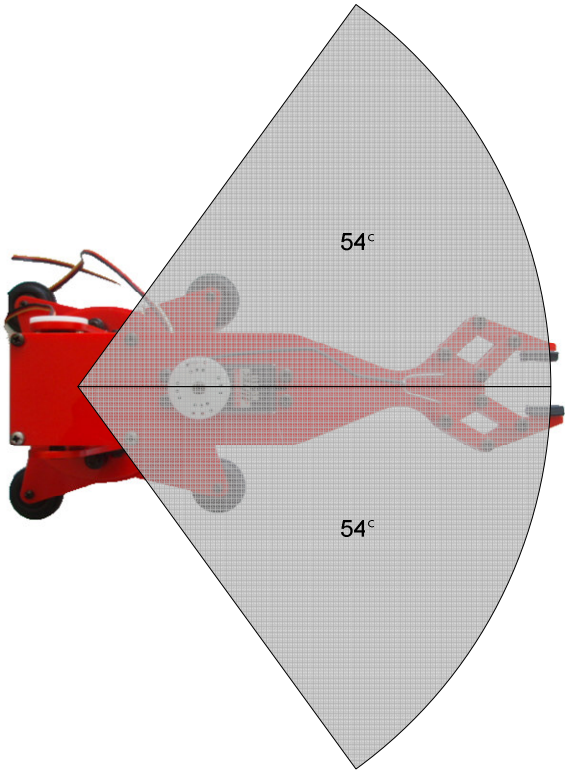
Pracovní prostor robota:

Otáčení základny: **± 54°**

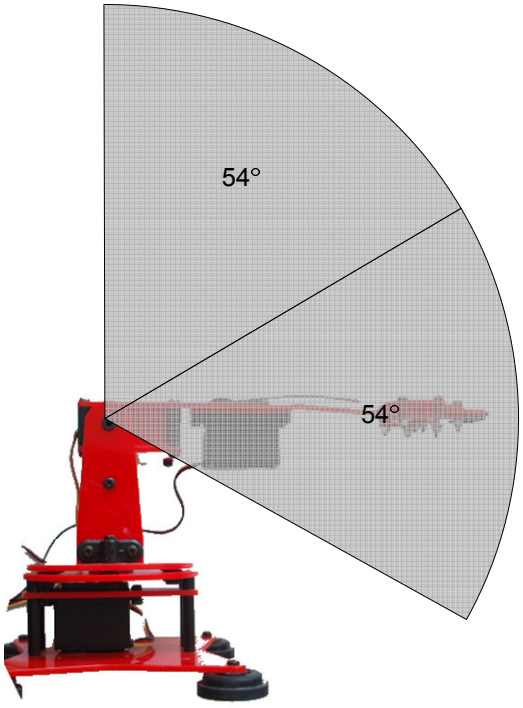
Zdvih ramene: **± 54°**

Pohyb kleštiny : **0- 35mm**

Obrázek 6.1: Rozsah pracovního prostoru kleštiny



Obrázek 6.2: Rozsah pracovního prostoru - půdorys



Obrázek 6.3: Rozsah pracovního prostoru - bokorys

7 Závěr

Studiem znalostí o rozhraní analogových joysticků, řízení modelářských servomechanismů a možností mikrokontrolérů v zastoupení rodiny HC08 jsem navrhl dvě zapojení modulů pro ovládání robotického manipulátoru, jednak s ohledem na maximální využití minimálního řešení, a dále s ohledem na budoucí možné využití a ladění ve školních laboratořích. Současné návrhy vzájemného propojení by měly umožnit dalšímu postupu práce v oblasti analýzy, návrhu a implementace různých zdrojových kódů určené pro tyto mikrokontroléry a ověření jejich funkčnosti na realizovaném prototypu.

Přestože je práce specializována na mikrokontroléry řady HC08, bylo by možno stejně dobře využít mikrokontroléry jiných výrobců, ale nebyla by zajištěna přenositelnost programových implementací mezi kitem a mikrokontrolérem.

Literatura

[1] Beta Control: Starter Kit LJ12EVB, Brno, 2002.

[2] HC08.cz: Vývojový kit JANUS uživatelský manuál,
[online] <http://measure.feld.cvut.cz/groups/micro/HC08/soubory/qt4/janus-um.pdf> [2.1.2007]

[3] Hitec: HS-322HD, [online] <http://hitecrcd.com/Servos/hs322hd.htm> [2.1.2007]

[4] MINASI, M.: PC – velký průvodce hardwarem, Grada Publishing, 1996.
ISBN 80-7169-178-X

[5] Motorola: MC68HC08 Microcontrollers MC68HC908QT datasheet,
[online] <http://measure.feld.cvut.cz/groups/micro/HC08/soubory/qt4/MC68HC908QYQT.pdf>
[2.1.2007]

[6] Motorola: MC68HC08 Microcontrollers MC68HC908LJ12 datasheet,
[online]
<http://www.freescale.com/files/microcontrollers/doc/data_sheet/MC68HC908LJ12.pdf>
[2.1.2007]

[7] NOVÁK, P.: Mobilní roboty – pohony, senzory, řízení, BEN-technická literatura, Praha, 2005,
ISBN 80-7300-141-1.

[8] Pinouts.ru team: PC Gameport (Joystick) pinout,
[online] <http://pinouts.ru/Input/GameportPC_pinout.shtml> [2.1.2007]

[9] SCHWARZ, J., RŮŽIČKA, R., STRNADEL, J.: Mikroprocesorové a vestavěné systémy,
[online]
<https://www.fit.vutbr.cz/study/courses/IMP/private/VYUKA/PREDNASKY/STUDIJNI_OPORA/IMP_opora_pr.pdf> [2.1.2007]

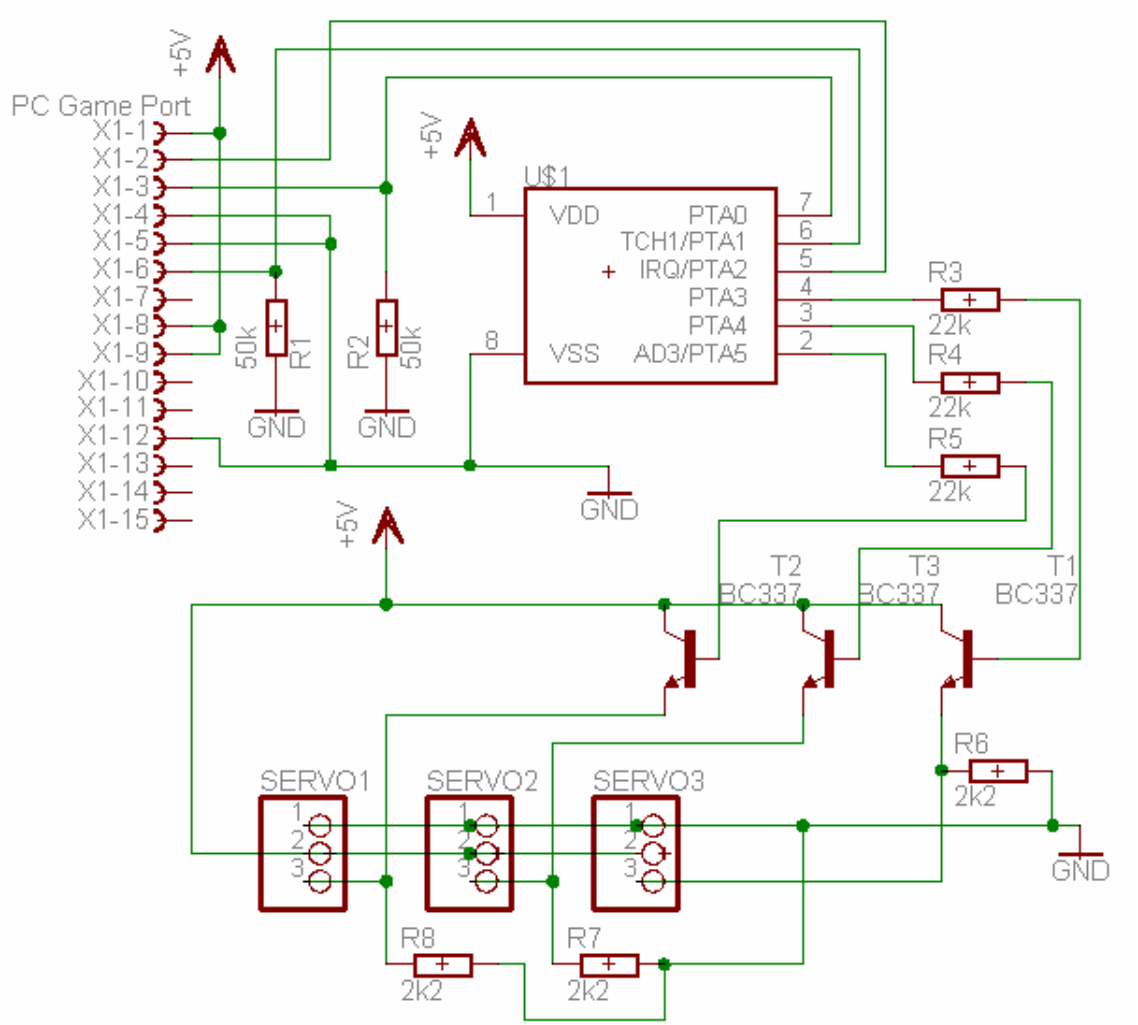
[10] SCHWARZ, J., DVOŘÁK, V., BUREŠ, P.: Číslicové a impulsové obvody, VUT v Brně, Brno,
1996, ISBN 80-7300-124-1.

[11] Siliconbrain: ROB 1-3 M, ROB 1-3 MS příručka ke stavebnici,
[online] <http://www.hobbyrobot.cz/PDF/ROB1-3_manual.pdf> [2.1.2007]

[12] VÁŇA, V.: Začínáme pracovat s mikrokontroléry MOTOROLA HC08 NITRON,
BEN-technická literatura, Praha, 2003, ISBN 80-7300-124-1.

Seznam příloh

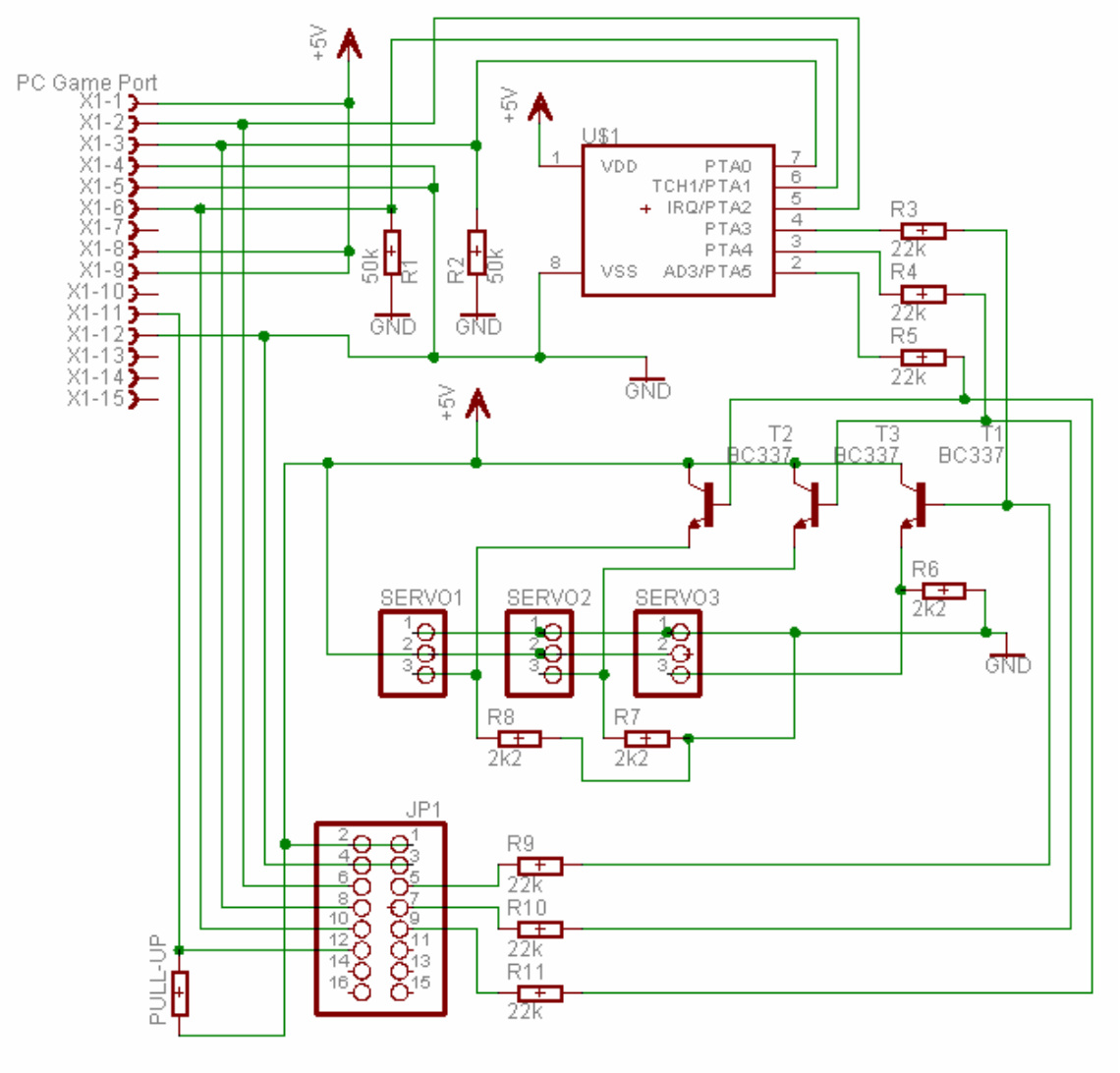
Příloha 1



Seznam použitých součástek:

- 1x mikrokontrolér **68HC908QT NITRON**
- 3x odpor **50kΩ**
- 3x odpor **22kΩ**
- 3x odpor **2k2Ω**
- 3x výkonový tranzistor **BC337**
- 1x konektor **PC Gameport**
- 3x konektor **PINHD-1X3**

Příloha 2



Seznam použitých součástek:

- 1x mikrokontrolér **68HC908QT NITRON**
- 3x odpor **50kΩ**
- 6x odpor **22kΩ**
- 3x odpor **2k2Ω**
- 3x výkonový tranzistor **BC337**
- 1x konektor **PC Gameport**
- 3x konektor **PINHD-1X3**
- 1x konektor **PINHD-2X8**

Příloha 3

```
/* Program ovládání robotického manipulatoru mikrokontrolerem NITRON */
/* autor:   Martin Zemanek                                           */
/* rocnik:  2MIT                                                     */
/* rok:     2007                                                     */
#include <hidef.h>      /* Finkce pro preruseni */
#include "derivative.h" /* Deklare periferii */
/*-------------------------------------------------------------------*/
/* Deklarace pouzitych promennych a funkci                           */
interrupt void TIMint();
void (*const obsluha)(void) @INT_TIMOvr = TIMint;

short servo[3]; /* delka pulsu jednotlivych serv vyjadrena jako prirustek hodnoty casovace  */
byte correction[43]={-84,-72,-60,-50,-40,-32,-25,-18,-12,-7,-3,
                      0,5,8,11,13,15,17,18,
                     20,20,21,22,22,22,22,22,
                     21,21,20,19,18,17,16,15,
                     13,12,10,9,7,5,3,1,}; /* korekcni pole hodnot */
byte trimjoystick[3]; /* stredni hodnota vychylek serv */
byte stav=9; /* stav automatu rizeni */
short minPuls=720; /* minimanlni delka pulsu pro servo, vzdy */
short period=13000; /* perioda pulsu */
/*-------------------------------------------------------------------*/
/* Procedura pro inicializaci servomechanicmu                        */
void initServos() {
 servo[0]=0;
 servo[1]=0;
 servo[2]=0;
 trimjoystick[0]=128;
 trimjoystick[1]=128;
 trimjoystick[2]=128;
}
/*-------------------------------------------------------------------*/
/* Inicilializace parametru casovace TIM                             */
void initTimer()
{
 TSC = 0x72; /* takt casovace 3.2MHz/4=800kHz */
 TMODH = 0x0; /* horni bajt casovace */
 TMODL = 0xFF; /* prvni preruseni az po 255 taktech */
 TSC = 0x42; /* spusteni casovace */
}
/*-------------------------------------------------------------------*/
/* Vypocet korekcni hodnoty z AD prevodniku                          */
byte correctAD(byte input)
{
 if (input<21)
  return 0;
 else
  return 4*input+correction[input-21];
```

```
}
/*---------------------------------------------------------------------*/
/* Funkce vypoctu novych parametru pro jednotlive n servo              */
short newServo(byte n)
{
  byte i;
  short pom=0;

  if (n<2)
    ADSCR = 0x20+n;     /* offset AD prevdniku na PTA + n (0/1) */
  if (~PTA&0x04)        /* tlacitko zmacknuto */
  {
   if (n<2)
    return servo[n];
   else
    ADSCR = 0x20;  /* offset AD prevdniku na PTA0 */
  }else if (n==2)
    return servo[n];

  for (i=0;i<15;i++); /* spozdeni pro evaluaci AD prvodu */
  pom=servo[n]+(correctAD(ADR/4)-trimjoystick[n])/20;  /* vypocet poctu taktu casovace pro
servo n */
  if (pom>960) pom=960;
  if (pom<0) pom=0;
  return pom;
}
/*---------------------------------------------------------------------*/
/* Procedura opracovani novych hodnot pro serva                        */
void setServos()
{
 byte i;
 for (i=0; i<3;i++)
  servo[i]=newServo(i);
}
/*---------------------------------------------------------------------*/
/* Funkce overi ze je na sbernici pripojen jiny mikrokontroler         */
byte isSlave()
 {
  byte i,j;
  for (i=0;i<252;i++)
   for (j=0;j<252;j++)
     if (PTA_PTA5) /* pokud je na vstupu signal na servo */
      return 1;
  return 0;
 }
/*---------------------------------------------------------------------*/
/* Nastaveni novehe  modulu hodnoty pro casovac                        */
void setDelay(short delay)
 {
   TSC = 0x72; /* takt 3.2MHz/4  */
   TMOD= delay;
```

```
   TSC = 0x42; /* spustit TIM */
 }
/*-------------------------------------------------------------------*/
/* Obsluha preruseni preteceni casovace                              */
interrupt void TIMint(void)
{

 if (stav==0)
 {
  /* pocatecni stav, nastupna hra na servu 1 + novy cas servo[0] */
          PTA=8;
          setDelay(minPuls+servo[0]);
          stav=1;
 } else if (stav==1)
 {
  /* nastupna hra na servu 2, sestupna na servu 1 + novy cas servo[1] */
          PTA=16;
          setDelay(minPuls+servo[1]);
          stav=2;
 } else if (stav==2)
 {
 /* nastupna hra na servu 3, sestupna na servu 2 + novy cas servo[2] */
           PTA=32;
          setDelay(minPuls+servo[2]);
          stav=3;
 } else if (stav==3)
 {
 /* sestupna na servu 3, vypocet zbyvajiciho casu do periody*/
          PTA=0;
          setDelay(period-servo[0]-servo[1]-servo[2]);
          stav=9;
 }
  TSC&=0x7F; /*vymaz priznaku preruseni*/
}
/*-------------------------------------------------------------------*/
/* Hlavni smycka programu                                            */
void main(void)
{
if (!isSlave())
 {
   CONFIG1=0x31;     /* zakazani COPD */
   CONFIG2=0;
   initServos();
   initTimer(); /* inicializace casovace */
   DDRA = 0x38; /* budou vystupy na serva */
   PTAPUE=0x04; /* tlacitko na PTA2 */
   /* povoleni preruseni */
   EnableInterrupts;
 }
for(;;) /* nekonecna smycka */
{
```

```
 /* stav automatu na zjisteni novych hodnot pro casovac */
 if (stav==9){
    setServos();
    stav=0;
 }
}
}
```

Příloha 4

```
/* Program ovládání robotického manipulatoru mikrokontrolerem LJ12   */
/* autor:   Martin Zemanek                                           */
/* rocnik:  2MIT                                                     */
/* rok:     2007                                                     */
#include <hidef.h>      /* Finkce pro preruseni */
#include "derivative.h" /* Deklare periferii */
/*-------------------------------------------------------------------*/
/* Deklarace pouzitych promennych a funkci                           */
interrupt void TIMint();
interrupt void DEMOint();
void (*const T1)(void) @INT_TIM1Ovr = TIMint;
void (*const T2)(void) @INT_TIM2Ovr = DEMOint;

short demo[8][3]; /* ulozene predchozi pozice */
short servo[3]; /* delka pulsu jednotlivych serv vyjadrena jako prirustek hodnoty casovace  */
byte correction[43]={-84,-72,-60,-50,-40,-32,-25,-18,-12,-7,-3,
                      0,5,8,11,13,15,17,18,
                     20,20,21,22,22,22,22,22,
                     21,21,20,19,18,17,16,15,
                     13,12,10,9,7,5,3,1,}; /* korekcni pole hodnot */
byte trimjoystick[3]; /* stredni hodnota vychylek serv */
byte stav=9; /* stav automatu rizeni */
byte actWr=0; /* ukazatel kam se bude ukladat pozice a cist */
byte actRd=0;
byte demos=0; /* demo=0 manualni rezim, demo=1 automaticky rezim */
short minPuls=720; /* minimanlni delka pulsu pro servo, vzdy */
short period=13000; /* perioda pulsu */

const unsigned char LCD1[20]=
{
    0x30, 0x35, /* display 0 */
    0x30, 0x00, /* display 1 */
    0x10, 0x27, /* display 2 */
    0x30, 0x07, /* display 3 */
    0x30, 0x12, /* display 4 */
    0x20, 0x17, /* display 5 */
    0x20, 0x37, /* display 6 */
    0x30, 0x01, /* display 7 */
    0x30, 0x37, /* display 8 */
    0x30, 0x13 /* display 9 */
};


const unsigned char LCD2[20]=
  {
    0x53, 0x03, /* display 0 */
    0x03, 0x00, /* display 1 */
    0x71, 0x02, /* display 2 */
```

```
    0x73, 0x00, /* display 3 */
    0x23, 0x01, /* display 4 */
    0x72, 0x01, /* display 5 */
    0x72, 0x03, /* display 6 */
    0x13, 0x00, /* display 7 */
    0x73, 0x03, /* display 8 */
    0x73, 0x01  /* display 9 */
  };

const unsigned char LCD3[20]=
{
    0x30, 0x35, /* display 0 */
    0x30, 0x00, /* display 1 */
    0x10, 0x27, /* display 2 */
    0x30, 0x07, /* display 3 */
    0x30, 0x12, /* display 4 */
    0x20, 0x17, /* display 5 */
    0x20, 0x37, /* display 6 */
    0x30, 0x01, /* display 7 */
    0x30, 0x37, /* display 8 */
    0x30, 0x13 /* display 9 */
};

const unsigned char LCD4[20]=
  {
    0x53, 0x03, /* display 0 */
    0x03, 0x00, /* display 1 */
    0x71, 0x02, /* display 2 */
    0x73, 0x00, /* display 3 */
    0x23, 0x01, /* display 4 */
    0x72, 0x01, /* display 5 */
    0x72, 0x03, /* display 6 */
    0x13, 0x00, /* display 7 */
    0x73, 0x03, /* display 8 */
    0x73, 0x01  /* display 9 */
  };

 const unsigned char LCD5[10]=
  {
    0xD7,  /* display 0 */
    0x06,  /* display 1 */
    0xE3,  /* display 2 */
    0xA7,  /* display 3 */
    0x36, /* display 4 */
    0xB5,  /* display 5 */
    0xF5,  /* display 6 */
    0x07,  /* display 7 */
    0xF7,  /* display 8 */
    0xB7  /* display 9 */
  };
```

```
const unsigned char LCD6[10]=
  {
    0xD7,  /* display 0 */
    0x06,  /* display 1 */
    0xE3,  /* display 2 */
    0xA7,  /* display 3 */
    0x36, /* display 4 */
    0xB5,  /* display 5 */
    0xF5,  /* display 6 */
    0x07,  /* display 7 */
    0xF7,  /* display 8 */
    0xB7  /* display 9 */
  };
/*---------------------------------------------------------------------*/
/* Procedura pro inicializaci servomechanicmu                          */
void initServos() {
 servo[0]=0;
 servo[1]=0;
 servo[2]=0;
 trimjoystick[0]=128;
 trimjoystick[1]=128;
 trimjoystick[2]=128;
}
/*---------------------------------------------------------------------*/
/* Inicilializace parametru casovace TIM                               */
void initTimer()
{
 T1SC = 0x71; /* takt casovace 2.4MHz/4=800kHz */
 T1MOD = period; /* pocatecni cekaci doba rovna delce periody */
 T1SC = 0x41; /* spusteni casovace */
}
/*---------------------------------------------------------------------*/
/* Vypocet korekcni hodnoty z AD prevodniku                            */
byte correctAD(byte input)
{
 if (input<21)
  return 0;
 else
  return 4*input+correction[input-21];

}

//nacteni hodnot z joysticku
short newServo(byte n)
{
  byte i;
  short pom=0;

  if (n<2)
    ADSCR = 0x20+n;     /* offset AD prevdniku na PTA + n (0/1) */
```

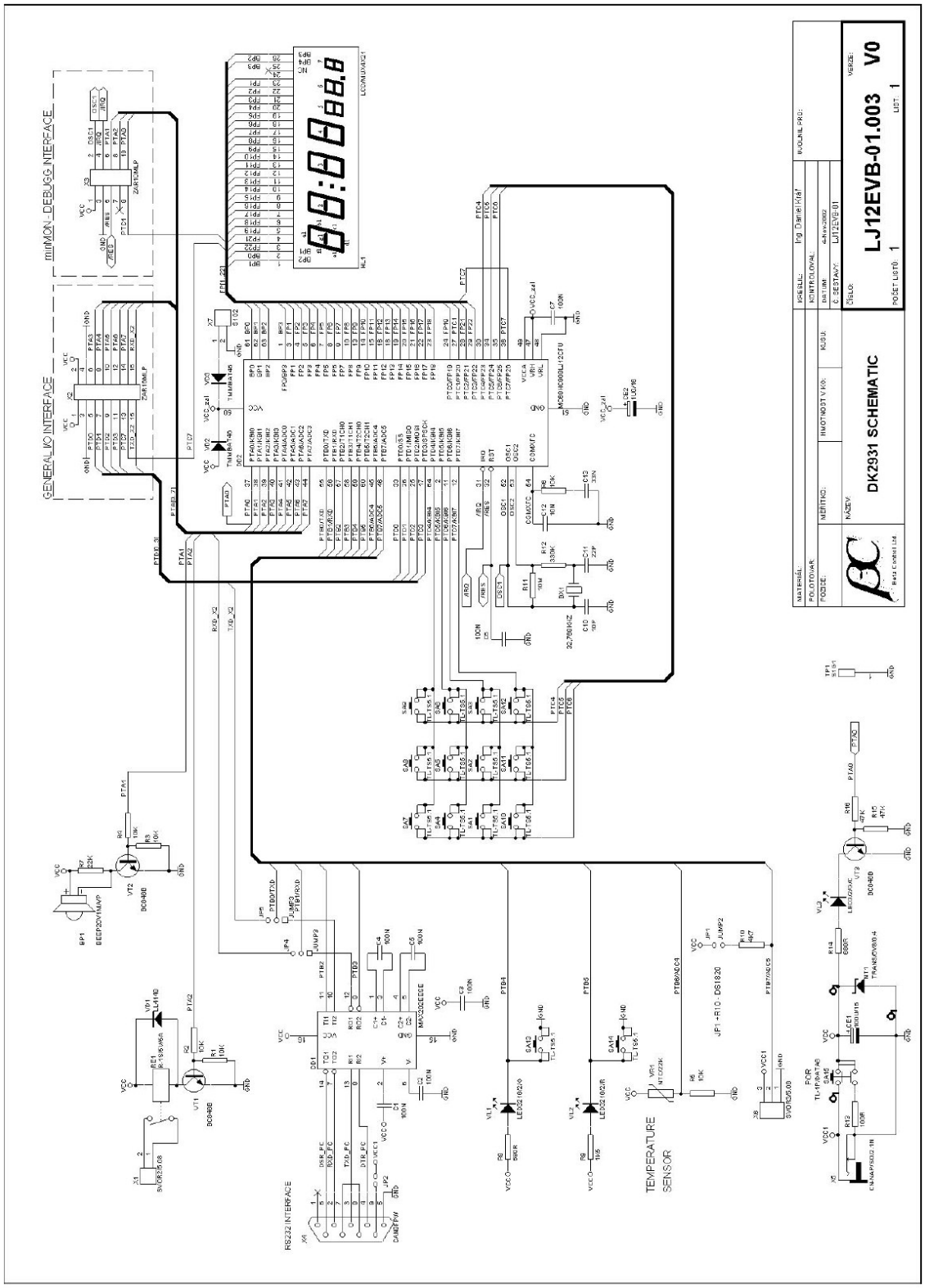
```
  if (~PTA&0x08)        /* tlacitko zmacknuto */
  {
   if (n<2)
    return servo[n];
   else
    ADSCR = 0x20; /* offset AD prevdniku na PTA0 */
  }else if (n==2)
    return servo[n];

  for (i=0;i<15;i++);  /* spozdeni pro evaluaci AD prvodu */
  pom=servo[n]+(correctAD(ADRL/4)-trimjoystick[n])/20;/*  vypocet  poctu  taktu  casovace  pro
servo n */
  if (pom>960) pom=960;
  if (pom<0) pom=0;
  return pom;

}
/*---------------------------------------------------------------------*/
/* Procedura zobrezeni aktualni pozice na displej                      */
void displayOnLCD()
{

  /* vymaz displeje */
  LDAT1=LDAT2=LDAT3=LDAT4=LDAT5=LDAT6=LDAT7=LDAT8=LDAT9=LDAT10=LDAT11=LDAT12=LDAT13=LDAT14=0;
  /* zobraz prvni servo 0 */
 LDAT10|=LCD1[2*(servo[0]*99/9600)];
 LDAT11|=LCD1[2*(servo[0]*99/9600)+1];

 LDAT9|=LCD2[2*((servo[0]/255*99)%10)];
 LDAT10|=LCD2[2*((servo[0]/255*99)%10)+1];

 /* zobraz druhe servo 1 */
 LDAT7|=LCD3[2*(servo[1]*99/9600)];
 LDAT8|=LCD3[2*(servo[1]*99/9600)+1];

 LDAT6|=LCD4[2*((servo[1]*99/9600)%10)];
 LDAT7|=LCD4[2*((servo[1]*99/9600)%10)+1];

  /* zobraz druhe servo 3 */
 LDAT5|=LCD5[(servo[2]*99/9600)];

 LDAT4|=LCD6[((servo[2]*99/9600)%10)];
}
/*---------------------------------------------------------------------*/
/* Kontrola stisku tlacitka a ulozeni hodnoty, zapnuti,vypnuti prehravani*/
void checkKB()
{
 byte button1,button2,button3;
 button1=PTD & 10;  /* vymaskovani tlacitek */
 button2=PTD & 20;
 button3=PTD & 40;
```

```
 if (button1) {      /* pridani dalsi pozice */
  if (actWr>7) actWr=0;
  demo[actWr][0]=servo[0];
  demo[actWr][1]=servo[1];
  demo[actWr][2]=servo[2];
  actWr++;
 }

 if (button2) {       /* prehravani */
  T2SC = 0x76; /* takt casovace 2.4MHz/64=37kHz */
  T2MOD = 0xFFFF; /* horni bajt casovace */
  T2SC = 0x46; /* spusteni casovace */
  demos=1;
 }
 if (button3) {       /* vypnuti casovace */
  T2SC = 0x76; /* takt casovace 2.4MHz/64=37kHz */
  demos=0;
 }
}
/*---------------------------------------------------------------------*/
/* Procedura opracovani novych hodnot pro serva                        */
void setServos()
{
 byte i;
 if (demos) return; /* automaticky rezim */
 for (i=0; i<3;i++)
  servo[i]=newServo(i);

}
/*---------------------------------------------------------------------*/
/* Nastaveni novehe  modulu hodnoty pro casovac                        */
void setDelay(short delay)
 {
   T1SC = 0x71; /* takt 2.4MHz/2=1.2MHz  */
   T1MOD= delay;
   T1SC = 0x41; /* spustit TIM */
 }
/*---------------------------------------------------------------------*/
/* Obsluha preruseni preteceni casovace TIM1                           */
interrupt void TIMint(void)
{

 if (stav==0)
 {
  /* pocatecni stav, nastupna hra na servu 1 + novy cas servo[0] */
          PTD=1;
          setDelay((minPuls+servo[0])*3/2);     /* generovani signalu je 1.5 rychlejis nez
NITRON */
          stav=1;
```

```
 } else if (stav==1)
 {
  /* nastupna hra na servu 2, sestupna na servu 1 + novy cas servo[1] */
          PTD=2;
          setDelay((minPuls+servo[1])*3/2);
          stav=2;
 } else if (stav==2)
 {
 /* nastupna hra na servu 3, sestupna na servu 2 + novy cas servo[2] */
           PTD=4;
          setDelay((minPuls+servo[2])*3/2);
          stav=3;
 } else if (stav==3)
 {
 /* sestupna na servu 3, vypocet zbyvajiciho casu do periody*/
          PTD=0;
          setDelay((period-servo[0]-servo[1]-servo[2])*3/2);
          stav=9;
 }
  T1SC&=0x7F; /*vymaz priznaku preruseni*/
}
/*-------------------------------------------------------------------------*/
/* Obsluha preruseni preteceni casovace TIM2                               */
interrupt void DEMOint(void) {

  if (actRd>7) actRd=0;
  servo[0]=demo[actRd][0];
  servo[1]=demo[actRd][1];
  servo[2]=demo[actRd][2];
  actRd++;

 T2SC&=0x7F; /*vymaz priznaku preruseni*/
}
/*-------------------------------------------------------------------------*/
/* Hlavni smycka programu                                                  */
void main(void)
{

   CONFIG1=0x31;     /* zakazani COPD */
   CONFIG2|=4;       /* povoleni LCD */
   initServos();
   initTimer(); /* inicializace casovace */
   DDRA = 0;    /* vstupy joysticku */
   DDRC = 0x40; /* tlacitka na PTC6 =1,7,10 */
   DDRD = 0x0F; /* vystupy na serva , vstupy z tlacitek */
   /* LCD */
   LCDCR=0xA5; /* parametry pro integrovany LCD */
   LCDCLK=0x53;
   /* KB */
   KBSCR = 0x02; /* low level, povoleni pullup */
   KBIER = 0x78; /* KBI3-6*/
```

```
   PTC= 0x30;
   /* ADR */
   ADCLK=0;
   /* povoleni preruseni */
   EnableInterrupts;
for(;;) /* nekonecna smycka */
{
 /* stav automatu na zjisteni novych hodnot pro casovac */
 if (stav==9){
    setServos();
    displayOnLCD();
    checkKB();
    stav=0;
 }
}
}
```

Příloha 5


LJ12EVB-01.003 V0
DK2931 SCHEMATIC
Ing.Daniel Kral
LJ12EVB-01
TEMPERATURE SENSOR
RS232 INTERFACE
GENERAL I/O INTERFACE
miniMON-DEBUGG INTERFACE